Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN SISTEMAS DE INFORMACIÓN

Trabajo Fin de Grado

ARQUITECTURA BIG DATA UTILIZANDO LAS HERRAMIENTAS NIFI, KAFKA Y ZEPPELIN

Autor: Alberto Moraga Fernández

Tutor/es: Iván González Diego

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN SISTEMAS DE INFORMACIÓN

Trabajo Fin de Grado

ARQUITECTURA BIG DATA UTILIZANDO LAS HERRAMIENTAS NIFI, KAFKA Y ZEPPELIN

Autor: Alberto Moraga Fernández

Tutor/es: Iván González Diego

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

FECHA:

Índice

RESUMEN

Cuando se empieza a indagar en el mundo del Big Data, lo que quizá se podía esperar encontrar es un conjunto acotado de tecnologías. La realidad es muy diferente y por ello una de las cosas más complejas es aprender a situarse y tener criterio. En este proyecto se va construir, implementar y configurar una arquitectura Big Data en la que se va a conseguir el tratamiento del dato al completo. A partir de información proveniente de la API de Twitter, se va a realizar la ingesta, almacenamiento, procesamiento y explotación del dato, utilizando para ello herramientas y tecnologías Big Data punteras en el sector.

RESUMEN EN INGLÉS

When you start to investigate in the world of Big Data, what you might expect to find is a limited set of technologies. The reality is very different and therefore one of the most complex things is to learn to situate yourself and have good judgment. In this project, a Big Data architecture will be build, implemented and configured in which the complete data treatment will be achieved. Based on information from the Twitter API, the data will be ingested, stored, processed and exploited, using top Big Data tools and technologies.

PALABRAS CLAVE

- Big Data
- Nifi
- Kafka
- Zeppelin
- Spark

INTRODUCCIÓN

MOTIVACIÓN PARA ELEGIR EL TEMA

Debido a la amplitud de temas y conceptos que se abordan a lo largo de los diferentes cursos del grado, la elección del tema a tratar en el TFG no ha sido tarea fácil. A la hora de elegir el tema a desarrollar en el TFG, quería algo que reflejase la formación que he ido adquiriendo durante el trascurso del grado, pero a su vez desarrollar, investigar y estudiar unos conceptos que me aportasen unos conocimientos que significasen un factor diferencial a la hora de poseer unas habilidades muy importantes para mi futuro laboral. Teniendo en cuenta estos factores, lo que más me llama la atención de las TIC es que cómo de un conjunto de datos inicial se puede llegar a sacar información, utilidades y conclusiones que no imaginaríamos poder sacar inicialmente. Por ello he decidido hacer un trabajo práctico de Big Data

En la sociedad actual el ser humano genera datos casi con cada acción que realiza. Esa generación de datos se está incrementando de forma exponencial y la tendencia es que ese crecimiento va a continuar, se prevé que en 2018 se generen 600 Zettabytes de datos (cerca de un billón de Terabytes). Pero los datos por sí solos no tienen ningún valor y por ello, se necesita un tratamiento del dato para ser capaces de obtener información y conocimiento.

El dicho de la información es poder, se lleva a su máximo nivel en las empresas. La capacidad de extraer valor de la información de una organización dictará su éxito en el futuro. El Big Data es el camino para poder extraer el máximo valor de todos los datos generados con la máxima velocidad y el mínimo coste. La correcta implementación del Big Data dentro de las empresas es cada vez menos un factor diferencial sino que se está convirtiendo en una cuestión de necesidad.

En definitiva es el tema que me parece más interesante del que queda mucho camino por recorrer, que está en plena actualidad, donde la evolución y perfeccionamiento del sector puede aportar grandes mejoras en nuestra sociedad. Por otro lado, uno de los grandes objetivos que perseguía es que el contenido de mi trabajo no estuviese previamente desarrollado, pudiendo aportar información verdaderamente valiosa.

INTRODUCCIÓN BIG DATA

Para hacerse una idea de la importancia que tiene hoy en día el tratamiento del dato, se va a analizar algunos números de lo que pasa en internet actualmente

Ilustración 1 - 60 segundos en el universo online

Como se puede ver las cifras de las acciones que se realizan en internet son enormes. Todas y cada una de esas iteraciones que se realizan quedan registradas. Eso significa ingentes cantidades de datos que, o bien puede suponer un problema diario para las compañías o por el contrario, si se aplican correctamente las tecnologías adecuadas actuando proactivamente, los datos se convertirán en nuevos activos intangibles de crecimiento.

Una vez que ya se ha comentado la importancia que supone el correcto tratamiento de los datos, se va a introducir a qué nos referimos exactamente con el Big Data.

No existe una definición única para este concepto pero la que más encaja con el Big Data actual es la siguiente: Conjunto de tecnologías que permiten a las organizaciones almacenar y analizar ingentes volúmenes de información con bajos tiempos de respuesta.

El Big Data surge de nuevas tecnologías, que permiten mediante nuevos frameworks, el tratamiento distribuido de grandes cantidades de datos, trabajando con miles de máquinas de forma distribuida, ampliando las limitaciones actuales y nuevas necesidades sobre capacidades de los sistema de Business Intelligence tradicionales. En definitiva, el Big Data ofrece frente a los sistemas tradicionales un menor coste, una alta escabilidad, un mayor rendimiento, una alta eficiencia y confiabilidad.

Principales características

Almacenamiento distribuido. Sistema de ficheros distribuido en diferentes máquinas mediante una estructura basada en nodos independientes, consiguiendo así, una alta escalabilidad. Además este almacenamiento es tolerante a fallos mediante datos replicados para soportar la caída de algún nodo.

Procesado en paralelo. Los procesos se dividen entre los diferentes nodos, trabajando en paralelo y consiguiendo un alto rendimiento. El escalado en paralelo de los diferentes nodos es sencillo y permite una mayor potencia del sistema. Este procesado de forma paralela permite realizar procesado en tiempo real.

Información diversa y no estructurada. Los datos con lo que se trabaja son de un gran volumen y de orígenes y formatos muy diversos. Además de los datos estructurados, que es la información que se ajusta a la estructura formal de los modelos de datos relacionales, los datos semi-estructurados, que son aquellos que no se ajustan a la estructura formal de los modelos relacionales, pero contienen etiquetas u otros marcadores para separar elementos semánticos (por ejemplo un XML); el Big Data es capaz de trabajar con datos no estructurados que son difíciles de tratar mediante el uso de programas informáticos tradicionales. Esta información no tiene un modelo predefinido de datos y/o no encaja bien en tablas relacionales. Algunos ejemplos de datos no estructurados pueden ser libros, revistas, audio, video o el cuerpo de un correo electrónico.

Las 7 V´s del Big Data

Para definir los conceptos más importantes del Big Data, se utiliza comúnmente las llamadas 7 V´s del Big Data.

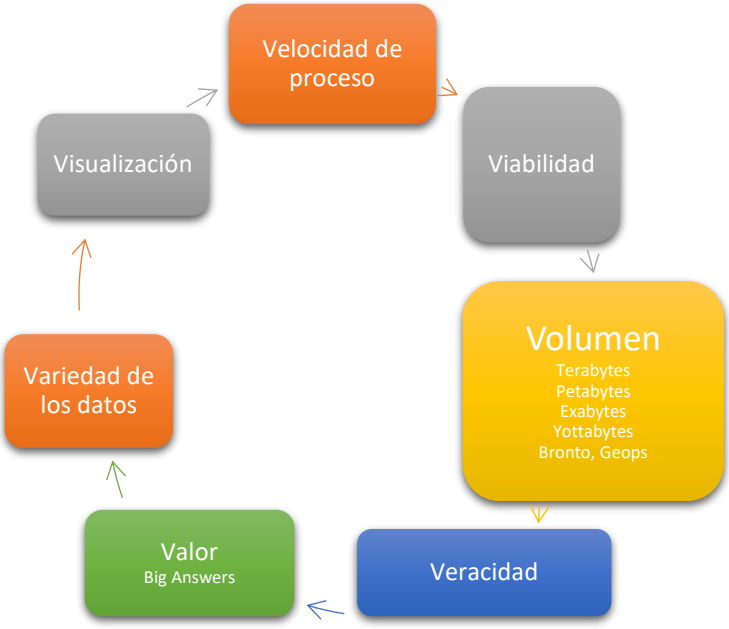


Ilustración 2 - Las 7 V's del Big Data

Volumen: Cantidad de datos que se generan cada segundo, minuto y días en nuestro entorno. Es la característica que el público común más asocia al Big Data, ya que hace referencia a las cantidades masivas de datos que se almacenan con la finalidad de procesar dicha información.

Velocidad: Hace referencia a la rapidez en las que los datos son creados, almacenados y procesado en tiempo real. Los datos están siempre en movimiento por las constantes interacciones que se realizan.

Variedad: Son las diferentes formas, tipos y fuentes en las que se registran los datos. Cómo ya se ha explicado anteriormente, los datos pueden ser datos estructurados como por ejemplo las bases de datos, que son los más fáciles de gestionar; pueden ser semi-estructurados o no estructurados. Entre los datos estructurados se encuentran por ejemplo textos, correos electrónicos, datos de sensores o publicaciones de redes sociales. Los no estructurados requieren de herramientas específicas cualificadas.

Veracidad: Es el grado de fiabilidad de la información recibida. Cuando nos referimos a este término, se hace referencia a la incertidumbre de los datos. Es importante invertir el tiempo necesario para conseguir datos de calidad.

Valor: Los datos por sí solos no tienen ningún valor. El valor se obtiene de datos que mediante la utilización del Big Data somos capaces de transformar en información; este a su vez en conocimiento y gracias a ese conocimiento adquirido poder tomar las mejores acciones que serán determinantes para el futuro de esa empresa.

Viabilidad: Es la capacidad que tienen las compañías de hacer un uso eficaz el gran volumen de datos que manejan. La inteligencia empresarial es muy importante para el éxito de un proyecto y su viabilidad. Una empresa con inteligencia competitiva es aquella que realiza innovación de los equipos de trabajo y un continuo análisis, estudio y aprendizaje de las tecnologías que van surgiendo en el mercado.

Visualización: Es el modo en que los datos son presentados. Una vez que los datos han sido procesados, necesitamos que los resultados se muestren visualmente de la mejor forma para que sean accesibles y legibles, consiguiendo ser explotados correctamente. Como ejemplo, se pueden visualizar en forma de dashboards y widgets.

INTRODUCCIÓN AL PROBLEMA A RESOLVER. OBJETIVOS

Cuando se empieza a indagar en el mundo del Big Data, lo que quizá se podía esperar encontrar es un conjunto acotado de tecnologías o casos de uso claros. La realidad es muy diferente, ya que existen múltiples herramientas. Esto es lo que se puede encontrar en la actualidad.

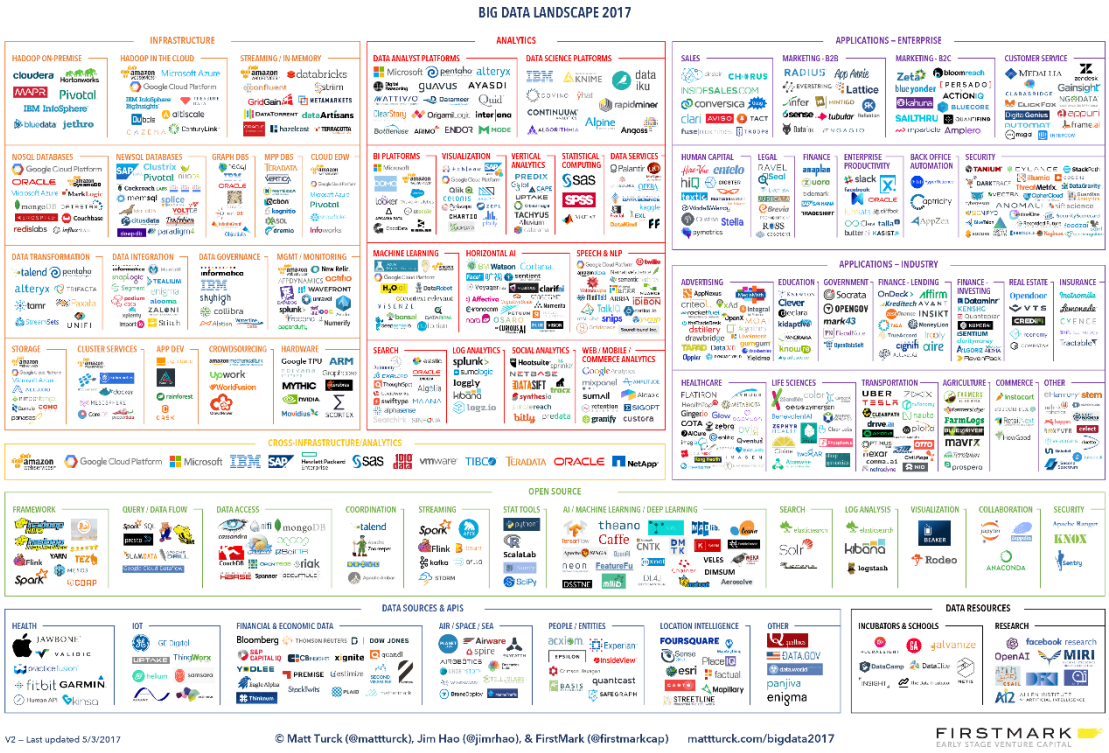


Ilustración 3 - Big Data landscape 2017

Como se puede observar, dada la cantidad de tecnologías existentes, una de las cosas más complejas en Big Data es aprender a situarse y conseguir esa foto general, en otras palabras, a tener criterio.

Algunas empresas como Hortonworks o Cloudera ofrecen distribuciones completas con algunas de estas herramientas. Estas distribuciones son una buena solución para algunos casos, pero el problema de estos productos es que hay que adaptarse a su producto y tienen poca flexibilidad.

Ante esta problemática, se decide crear nuestro propio entorno Big Data con las herramientas más convenientes. Una arquitectura Big Data está formada, a grandes rasgos, por herramientas para la ingesta de datos, el almacenamiento, el procesamiento y la visualización de los datos.

La opción de crear nuestro propio entorno Big Data, hace que la complejidad sea mucho mayor que la de optar por utilizar una distribución ya creada o la de realizar un trabajo que utilice una única herramienta de procesamiento de datos, o de ingesta, o de almacenamiento o visualizar unos datos ya existentes. El objetivo es la realización y configuración de una arquitectura Big Data completa de principio a fin, siendo capaz de llegar desde la ingesta de datos hasta la visualización para una correcta explotación de los datos, pasando por su almacenamiento y procesamiento. De esta manera conseguiremos reflejar correctamente lo que es el Big Data.

CASO DE USO Y ARQUITECTURA ELEGIDA

En las redes sociales se crean miles de contenidos diferentes por segundo. Todas las personas que disponen de una conexión a internet tienen la posibilidad de crear ese contenido y se trata de la forma más fácil en las que las personas de a pie pueden generar información propia libremente, que bien utilizada puede ser muy valiosa. Se quiere aprovechar ese potencial que ofrecen los datos de las redes sociales. Con ese objetivo se va a realizar nuestro entorno Big Data

La red social elegida es Twitter ya que se considera que el Dataset que puede proporcionar esta aplicación es la que tiene más potencial a la hora de realizar un procesamiento de los datos para su explotación. Esto es debido a que además de la información en forma de texto, foto o vídeo que incorpora el usuario, el tweet lleva asociado más información que muchas veces los usuarios no consideran, como puede ser la geolocalización o si se trata de un retweet lleva consigo toda la información del usuario que ha creado el tweet original, etc.

Nuestro objetivo es una vez que se consigan obtener los datos de la Api de Twitter, almacenarlos, realizar un procesamiento Spark mediante las diferentes herramientas y finalmente poder visualizar los datos una vez procesados de la mejor forma para sacar las mejores conclusiones posibles y poder sacar el mayor rendimiento y beneficio, obteniendo unos conocimientos que serían imposible de obtener con los datos iniciales en bruto

Para conseguir este propósito la arquitectura que se ha propuesto inicialmente es la siguiente:

Figura 1 - Propuesta de arquitectura inicial

Sin embargo, una vez que nos hemos adentrado en el desarrollo e implementación de la arquitectura, se ha descubierto una herramienta que permite realizar procesamiento en streaming. La herramienta en cuestión se denomina Kafka Streams y permite realizar un procesamiento en tiempo real de la salida del topic de Kafka.

Con la introducción de Kafka Streams la arquitectura final es la siguiente:

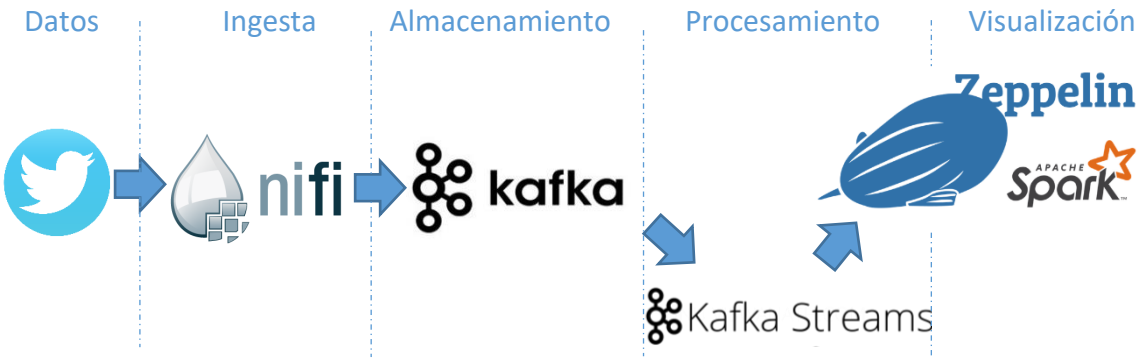


Figura 2 - Arquitectura final

Para la captura de los datos se utilizará Nifi. Apache Nifi es una herramienta que es capaz de mover datos entre cualquier fuente y cualquier destino, automatizando y gestionando el flujo de datos entre sistemas. La información que se ingesta en Nifi se irá almacenando en Kafka. Apache Kafka es un sistema de almacenamiento productor (producer), consumidor (consumer) que funciona como un servicio de mensajería, categorizando los mensajes en topics. Posteriormente para preparar el archivo del cual poder crear el RDD, para su posterior procesamiento en Zeppelin, se realizará un programa donde se implementará Kafka Streams. El entorno de programación elegido es Eclipse y será programado en Scala, ya que es el que mejor rendimiento obtiene. Finalmente se realizará la explotación de los datos en Zeppelin. Apache Zeppelin es una implementación del concepto de web notebook, centrado en la analítica de datos interactivo mediante lenguajes y tecnologías como Shell, Spark, SparkSQL, etc. Por lo tanto, dentro del mismo notebook de Zeppelin se podrá hacer procesamiento y visualización.

HERRAMIENTAS UTILIZADAS

APACHE NIFI

- Developer(s): National Security Agency,
- Onyara, Inc., Hortonworks
- Stable release: 1.6.0
- Written in: Java
- Type: Distributed dataflow
- License: Apache License 2.0
- Website: http://nifi.apache.org/

¿Qué es Apache Nifi?

Es una aplicación con interfaz de usuario basada en web que permite automatizar el flujo de datos entre sistemas. Fue desarrollado por la NSA durante más de ocho años, hasta que en noviembre de 2014 fue donado a Apache Foundation.

Aunque el término flujo de datos se usa en una variedad de contextos, aquí lo usa para referirse al flujo de información automatizado y administrado entre los sistemas. Esta problemática ha existido desde que las empresas tienen más de un sistema y algunos de los sistemas creaban datos y otros sistemas consumían datos. Nifi ha sido diseñado para abordar estos desafíos de los flujos de datos.

A modos resumen, estos serían los puntos clave de Apache Nifi:

- Aplicación con interfaz de usuario basada en web
- Automatizar el flujo de datos entre sistemas
- Sistema basado en flujos
- Potente y Fácil de usar
- Sistema fiable de procesamiento de datos
- Permite distribuir datos
- Soporta potentes y escalables gráficos dirigidos al enrutamiento de datos

Conceptos básicos

Flowfile: Representa cada objeto que se mueve a través del sistema por los diferentes processors. Nifi realiza un seguimiento de cada uno de ellos con un mapeo de sus pares de clave/valor y su contenido asociado.

Processor: Son los que realmente realizan el trabajo. Son los diferentes componentes por los que van pasando los flowfiles, que se visualizan en forma de cajitas dentro de la

zona de trabajo en la interfaz gráfica. Por ejemplo son los que realizan la combinación de datos, transformación o mediación entre sistemas.

Connection: son los encargados de unir los diferentes Processsors. Actúan como colas y permite que diferentes procesos interactúen a diferentes velocidades.

Flow Controller: Mantiene el conocimiento de cómo los procesos conectan y gestionan los subprocesos y actúa como intermediario facilitando el intercambio de FlowFiles entre procesadores.

Process Group: Conjunto de procesos y sus conexiones. Se puede crear nuevos componentes con uno o varios process groups. Pueden recibir datos a través de puertos de entrada y enviar datos a través de puertos de salida.

Processors

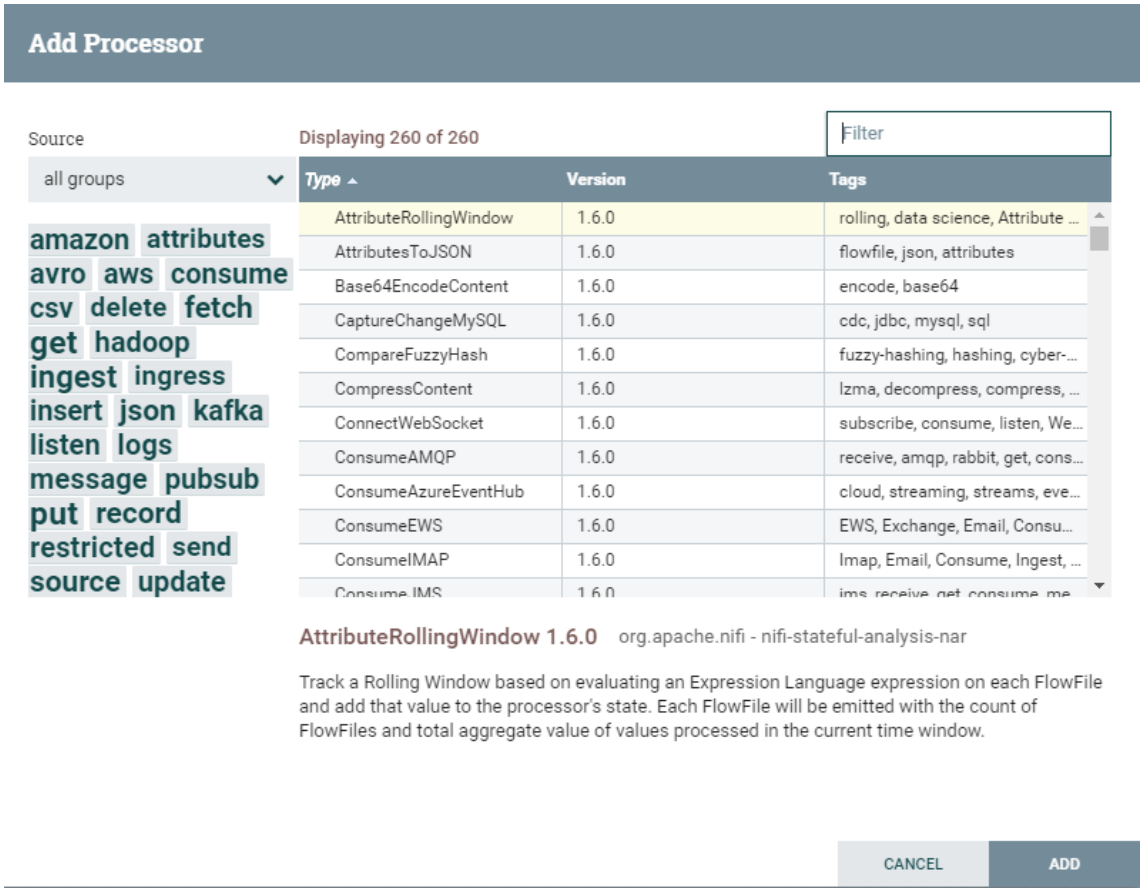


Ilustración 4 - Menú de añadir processors

- Cada una de las *cajas* que forman parte del flujo, los procesadores, aportan una funcionalidad concreta al flujo. NiFi viene pre-cargado con una larga lista de operaciones sobre los datos, en forma de procesadores:

- **Interacción con fuentes de datos**. Carga de datos desde, y hacia, una amplia variedad de formatos y tecnologías: HDFS, ElasticSearch, FTP, bases de datos SQL, MongoDB, ficheros… entre otros muchos.
- **Conversión de datos**. Cambios de estructura de la información, de y hacia JSON, XML, Avro, CSV, etc.
- **Proceso de la información**. Operaciones de unión y división de los datos, así como su validación y transformación.
- **Delegación de funcionalidades**. Procesadores que pueden traspasar datos a otros sistemas de procesamiento, para realizar tareas sobre ellos, que ya estén implementadas en otros sistemas: Por ejemplo, la comunicación bidireccional con un tercero, mediante colas Kafka, o la ejecución transparente de procesos de Apache Flume.

Para casos específicos en los que el tratamiento del dato requiera algo *ad hoc*, NiFi proporciona un mecanismo para integrar un proceso definido en una simple clase java, como un procesador más dentro del diseño del flujo.

Arquitectura

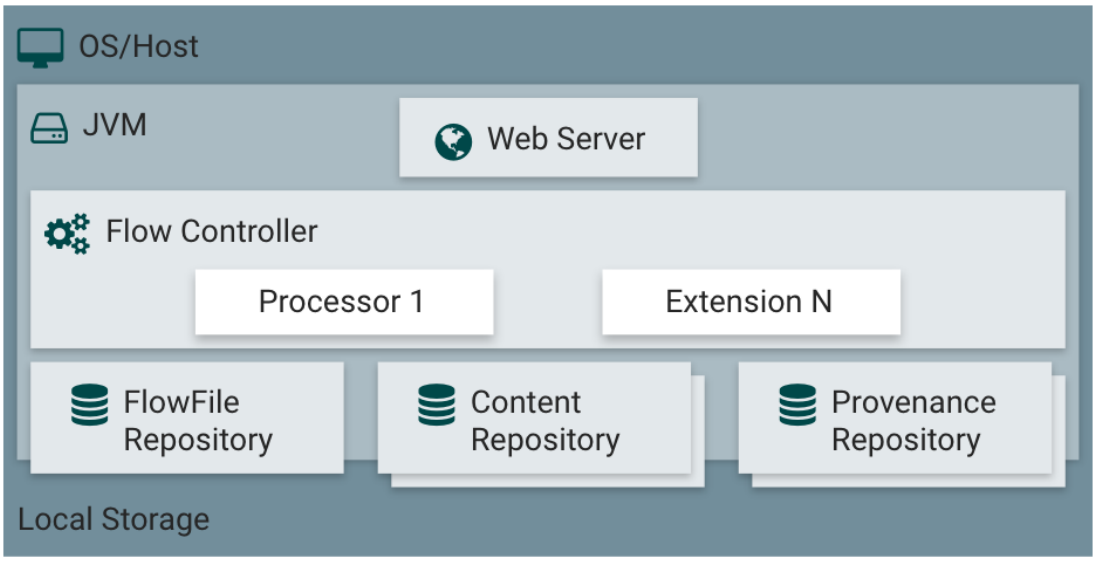


Ilustración 5 - Arquitectura Nifi

Interfaz de usuario

La aplicación web de Apache NiFi es uno de sus puntos fuertes. No sólo permite diseñar y configurar de forma visual el flujo por el que viajarán los datos, sino que, desde esa misma vista, se pueden tanto operar sobre el proceso (arranque y parada) como monitorizar el estado, viendo los posibles errores que se hayan producido.

Este flujo puede modificarse durante su ejecución, simplemente, deteniendo el proceso en un punto, y realizando los cambios. Los datos se encolan en el paso anterior, hasta que se reanude la ejecución.

Se puede:

- Construir un flujo arrastrando los processors.
- Iniciar, detener y configurar los componentes en tiempo real
- Ver errores en los mensajes
- Ver estadísticas y estado del flujo de datos
- Crear plantillas para conexiones comunes

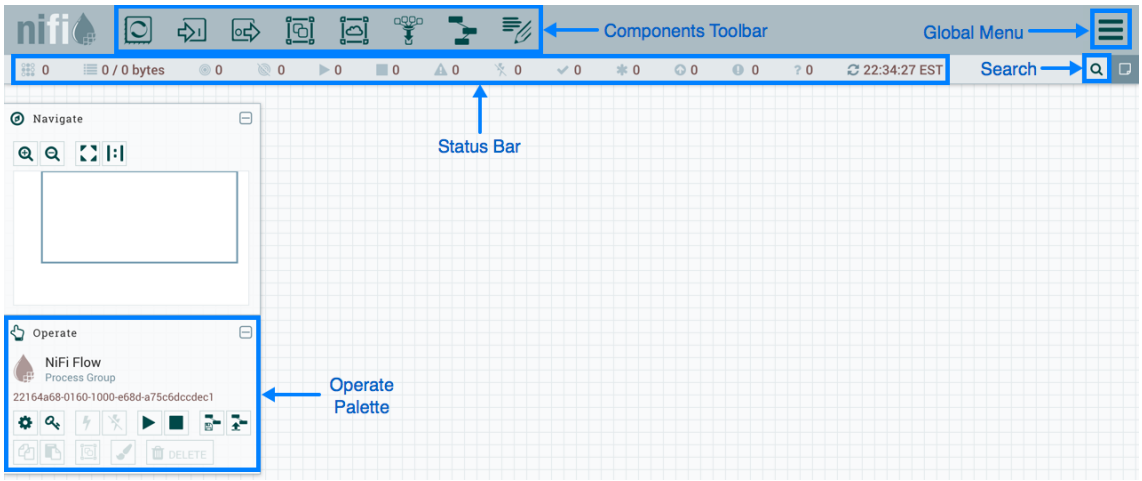


Ilustración 6 - Interfaz de inicio de Nifi

Características

Algunas de sus características entrando más al detalle son:

- Sistema potente y fiable para procesar y distribuir datos. Los datos se procesan en varios servidores antes de enviarlos al usuario o a un sistema de almacenamiento.
- Interfaz de usuario basada en web para la creación, seguimiento (diagramación visual) y control de los flujos de datos.
- Muy configurable a la hora de modificar el flujo de datos en tiempo de ejecución y dar prioridad a los datos de forma dinámica.
- Permite concurrencia, descomposición en diferentes componentes, pruebas y reutilización.
- Seguimiento de datos Origen a través de todo el sistema (clúster de servidores).
- Fácilmente extensible a través del desarrollo de componentes personalizados (JAVA).
- Escalable con nuevos procesadores, puede ser protegido por contraseña, y soporta conexiones seguras vía SSL, SSH y HTTPS.

Los problemas que se pueden evitar con Nifi son:

- Fallo de los sistemas.
- Cuello de botella de consumo en el flujo.
- Diferentes tamaños de datos y velocidades de sistemas.
- Seguridad en el flujo de datos entre sistemas.

Algunos ejemplos de casos de uso que resuelve Nifi son:

- Captura de la información de los sensores remotos.
- Flujos de datos entre sistemas.
- Flujos de datos dentro de los propios sistemas.
- Ingestión de grandes volúmenes de datos.
- Procesamiento de datos (enriquecimiento, la filtración, la desinfección).

APACHE KAFKA

- Developer(s): Apache Software Foundation
- Stable release: 1.1.0
- Written in: Scala
- Type: Cross-platform
- License: Apache License 2.0
- Website: kafka.apache.org/

¿Qué es Apache Kafka?

Apache Kafka es un sistema de almacenamiento publicador/subscriptor distribuido, particionado y replicado. Estas características, añadidas a que es muy rápido en lecturas y escrituras lo convierten en una herramienta excelente para comunicar streams de información que se generan a gran velocidad y que deben ser gestionados por uno o varias aplicaciones.

Es un proyecto opensource creado inicialmente por Linkedin para solventar las limitaciones de los sistemas de mensajería como Jms y finalmente Opensource en 2011.

Aunque luego se hablará más en detalle de ellos, se va a mencionar algunos de los conceptos de Kafka.

Topic: Kafka clasifica y mantiene los mensajes en categorías llamadas Topics.

Producer: Clientes conectados a Kafka responsables de publicar los mensajes. Estos mensajes son publicados sobre uno o varios topics.

Consumer: Los consumers están suscritos a uno o varios topics y reciben los mensajes publicados en estos topics. En otras palabras, son los encargados de consumir los mensajes del topic.

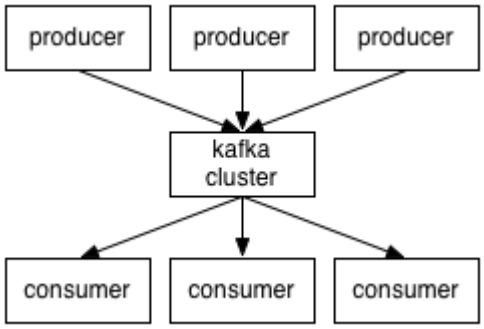


Ilustración 7 - Clientes Kafka

Broker: Los brokers son cada uno de los nodos de Kafka que forman el clúster.

Sistema de colas

Una cola de mensajes es un contenedor que alberga mensajes mientras están en tránsito. El principal objetivo de una cola es proporcionar enrutamiento y garantizar la entrega de los mensajes. Si el destinatario no está disponible en el momento de enviarse un mensaje, la cola lo guarda hasta que pueda entregarse correctamente.

Se diferencian principalmente dos tipos de sistemas de colas:

- Modelo Punto a Punto (point to point)

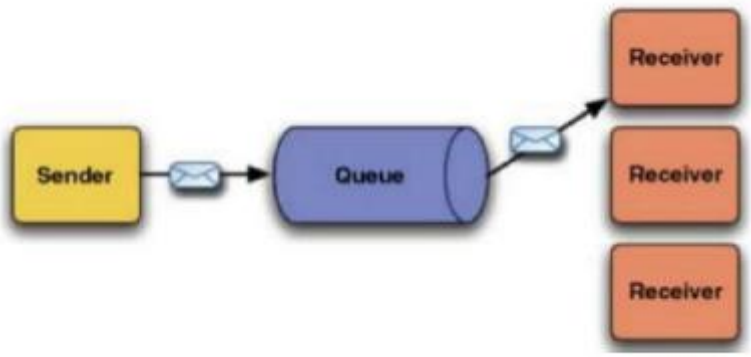


Ilustración 8 - Modelo de colas punto a punto

- Modelo Publicador / Consumidor (Publish/Suscribe). **Es el de Kafka**

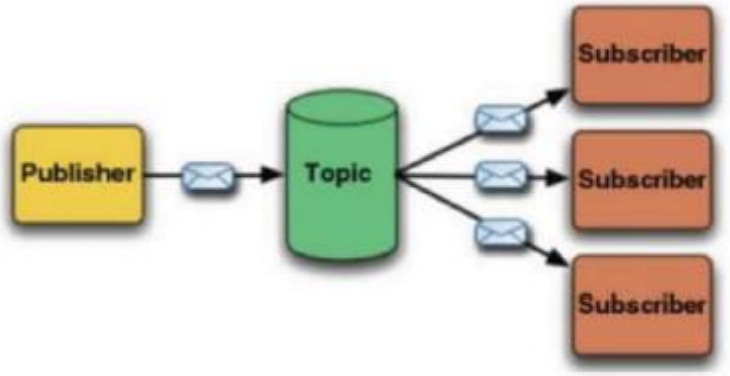


Ilustración 9 - Modelo de colas Publisher/Consumer

Protocolo de comunicación

La comunicación entre los clientes (consumers y producers) con el nodo de Kafka se realiza mediante un protocolo simple, de alto rendimiento y agnóstico; el protocolo TCP. Este protocolo está versionado pero mantiene la compatibilidad con versiones anteriores. Permite la creación e implementación de nuevos clientes realizados en cualquier lenguaje de programación, evitando las limitaciones de las interfaces y APIs definidas de JMS. Es el mismo concepto seguido por otros protocolos de mensajería, como es el caso de AMQP (Advanced Message Queuing Protocol).

Topics y logs

Un topic es una categoría o una cola donde publicar los mensajes. Los topic son multisuscriptor, es decir, cada topic puede tener cero, uno o varios consumers que están suscritos a un mismo topic.

Cada Topic Kafka los mantiene en un log particionado, cuya representación es la siguiente:

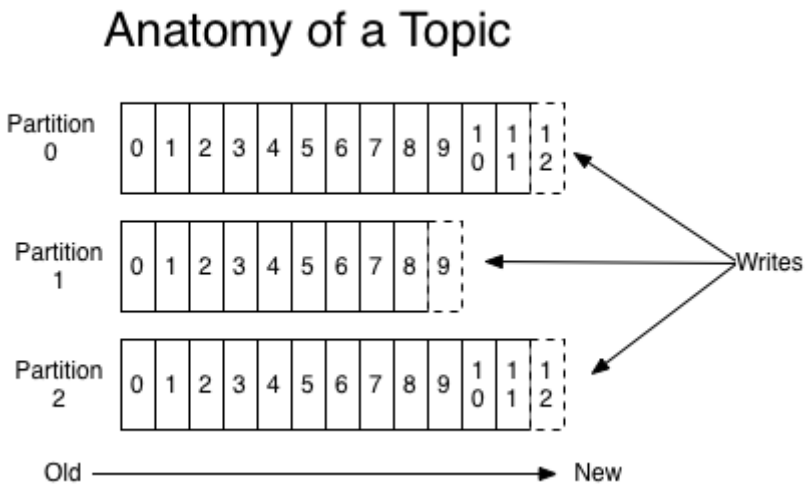


Ilustración 10 – Particiones de un Topic

Cada partición es una secuencia de mensajes inalterable, donde los mensajes son añadidos a una de las particiones según van llegando, y se les va asignando un número secuencial, llamado offset, que identifica de forma única a cada mensaje dentro de su partición.

Todos los mensajes son mantenidos en las particiones para su consumo durante un periodo de tiempo configurado en a nivel de clúster. Una vez se supera ese periodo de tiempo los mensajes son eliminados para liberar espacio.

Más tiempo pase, el tamaño de estas particiones crece, pero no afecta al rendimiento de Kafka.

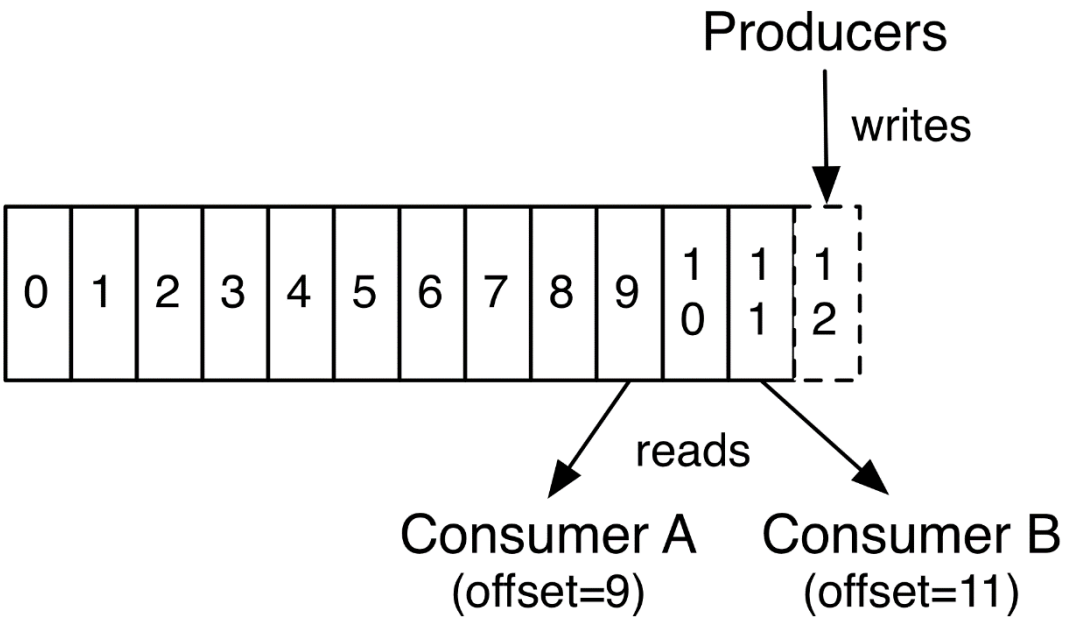


Ilustración 11 – Escritura y lectura del topic

El consumidor es responsable de mantener el offset del último mensaje consumido, no es mantenido por el clúster. Esto permite que un consumidor pueda consumir los mensajes en cualquier orden. Esto es otra de las ventajas que Kafka proporciona.

Gracias a poder dividir un Topic en un log de varias particiones, permite escalar de forma horizontal, creando, si fuera necesario, más particiones en el resto de nodos del clúster.

Sistema distribuido y Apache Zookeeper

Kafka proporciona tolerancia a fallos replicando cada partición del log en un número configurable de servidores dentro del clúster. La caída de uno de estos nodos no afecta al servicio. Uno de estos nodos actúa de leader mientras que el resto actúan de "followers". Este leader atiende todas las lecturas y escrituras de la partición mientras que los followers actúan como consumidores del leader replicando su contenido.

Si el leader falla automáticamente uno de los followers toma el rol del leader.

En un clúster de Kafka cada servidor actúa como leader de una partición y como follower de una partición diferente, por lo que la carga del sistema se encuentra bien balanceada en todo el clúster.

Apache Zookeeper

Zookeeper es una herramienta utilizada por muchas otras aplicaciones para almacenar ciertas configuraciones. Es capaz de almacenar espacios de nombres y configuraciones de forma distribuida, por lo que con él se obtiene tolerancia a fallos, alta disponibilidad, etc. En muchas herramientas es utilizado para poder restaurar estados concretos tras una caída o reinicio. Es decir, si una aplicación se para completamente (todos sus nodos), al reiniciarse será capaz de recuperar su último estado gracias a Zookeeper.

En Kafka sirve como interfaz de coordinación entre los kafka brokers y los Kafka consumers. Mantiene un registro de todos los topics y consumers que existen. Además, conoce qué mensajes ha consumido cada consumer. De esta forma, si un consumidor antiguo vuelve a conectarse, Zookeeper se encarga de conectar con Kafka y proporcionarle todos los mensajes que le faltan.

Consumer y Producers

Producers

Los producers publican mensajes en uno o varios topics. Son los responsables de elegir qué registro asignar a qué partición dentro del topic. Esto se puede hacer de forma "round-robin", es decir, de forma rutinaria simplemente para equilibrar la carga o se puede hacer siguiendo un criterio definido, como puede ser por algún tipo de algoritmo basado en una clave del mensaje.

Consumers

Kafka proporciona un único nivel de abstracción donde recoge el modelo de mensajería de *queuing* (colas) y *publish-subscribe* (editor-subscriptor). Este nivel de abstracción es llamado *consumer group* o grupo consumidor.

- Los consumidores se inscriben o subscriben a un grupo consumidor. Un grupo consumidor tendrá 1 a 'n' consumidores.

- Cada mensaje publicado en un topic es entregado solamente a uno de todos los consumidores que pertenecen a un mismo grupo consumidor. De esta forma se consigue la funcionalidad de un sistema de mensajería de colas.

- En el caso de que existan varios grupos de consumidores subscritos a un mismo topic, cada mensaje publicado en un topic es entregado a todos los grupos consumidores, y dentro de cada grupo consumidor a un solo consumidor. Esta es una de las formas de conseguir el modelo editor-subscriptor.

- Si todos los consumidores tienen diferente grupo consumidor, cada mensaje es enviado a todos los consumidores. Es el modelo editor-subcriptor.

- Si todos los consumidores tienen el mismo grupo consumidor cada mensaje es enviado a solo un consumidor del grupo de consumidores. Es el modelo de colas.

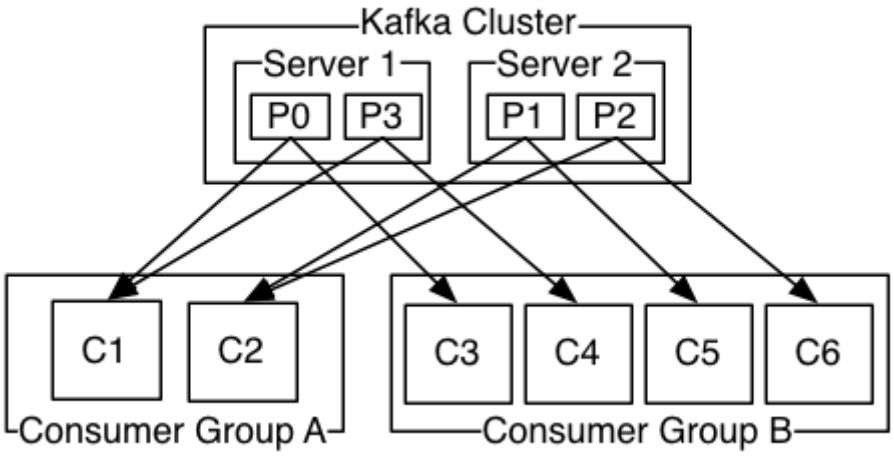


Ilustración 12 - Funcionamiento de los group consumer

Normalmente los sistemas de mensajería como Jms guardan y van entregando los mensajes en el mismo orden en el que llegan. Sin embargo una vez que llegan uno a uno al consumidor, el proceso del mensaje es asíncrono y por tanto el orden de salida del mensaje se puede llegar a perder. Para garantizar el orden en la entrega de los mensajes se asigna un único consumidor, lo que hace que se pierda la capacidad de escalado.

Kafka aporta una solución a este problema, permitiendo el escalado y la garantía de la entrega de los mensajes en el orden en el que llegan.

Como sabemos, se puede dividir un topic en varias particiones (cada partición podría estar replicada, para garantizar la tolerancia al fallo), con lo que se logra escalar el sistema. Los productores de mensajes pueden publicar los mensajes en base a una clave y por tanto son enviados a la misma partición. Finalmente a cada partición se subscribe un grupo consumidor, momento en el que solo un consumidor del grupo comienza a procesar los mensajes de esa partición, mientras se procesa también el resto de mensajes de las otras particiones consumidos por otros consumidores, logrando garantizar el orden de entrega de los mensajes, mientras se sigue balanceando y escalando el sistema.

Solamente en el caso de no poder dividir el topic en varias particiones, o de no poder enviar los mensajes a la misma partición usando una clave, se volvería a tener la limitación de un único consumidor y se dejaría de escalar el sistema.

Kafka como sistema de almacenamiento

Cualquier cola de mensajes que permita publicar mensajes de forma desacoplada al consumidor está actuando como un sistema de almacenamiento para los mensajes en movimiento. Lo que le hace a Kafka diferenciarse del resto de sistemas de colas, es que es un muy buen sistema de almacenamiento.

Los datos escritos en Kafka se escriben en disco y se replican para asegurar la tolerancia a los fallos. Kafka no considera que se ha realizado la escritura completa por parte de un producer hasta que la escritura se haya replicado por completo y se haya garantizado que persistirá aunque el servidor haya fallado.

Las estructuras de disco que utiliza Kafka escalan de una forma envidiable ya que tendrán el mismo rendimiento si tiene 50 Kb o 50 Tb de datos persistentes en el servidor.

Como resultado de este buen almacenamiento y permitir que los clientes controlen su posición de lectura, se considera a Kafka como un tipo de sistema de archivos distribuidos con el propósito en especial del almacenamiento, la replicación y la propagación de registros. Todo ello con un alto rendimiento, baja latencia y con tolerancia a los fallos.

APACHE ZEPPELIN

- Developer(s): Apache Software Foundation
- Stable release: 0.7.3
- Type: Notebook web
- License: Apache License 2.0
- Website: zeppelin.apache.org/

Concepto Notebook

El concepto de “**notebook**” fue introducido por iPyyhon, que permitía trabajar sobre un interfaz web en lugar de sobre una Shell.

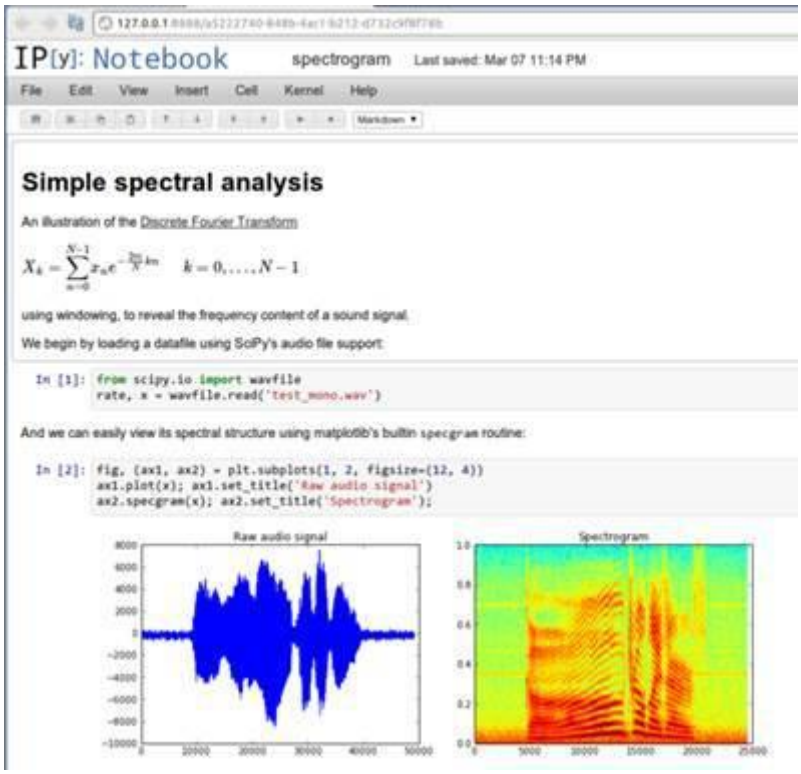


Ilustración 13 – Notebook de iPython

El notebook permite compartir tus procesos con otros, de modo que estos puedan entenderlos, modificarlos y adaptarlos a sus necesidades.

A correlación con este concepto se tiene el concepto del web notebook, que permite trabajar sobre una interfaz web como si de una Shell se tratase.

¿Qué es Apache Zeppelin?

Apache Zeppelin es una implementación del concepto de web notebook**,** centrado en la analítica de datos interactivo mediante lenguajes y tecnologías como Shell, Spark, SparkSQL, Hive, Elasticsearch, R, etc. Ofrece funciones de exploración, visualización, intercambio y colaboración de datos a Spark.

En la propia web de Zeppelin, su principal eslogan es que el Notebook es el lugar para todo lo que se necesita:

- Data Ingestion
- Data Discovery
- Data Analytics
- Data Visualization & Collaboration

Intérpretes de múltiples lenguajes

Apache Zeppelin interpreter permite que multitud de lenguajes y tipos de procesaiento de datos puedan ser añadido a Zeppellin. Actualmente Apache Zeppelin admite múltiples intérpretes como Apache Spark, Python, JDBC, Markdown and Shell.

Integación con Apache Spark

Apache Zeppelin provee especialmente una integración con Apache Spark, sin la necesida de crear un módulo por separado, plugin o librería para ello.

Apache Zeppelin con integración Spark proporciona:

- Implementación automatica de SparkContext y SQLContext.
- Carga la dependencia del jar en tiempo de ejecución desde el sistema de archivos lcal o el repositorios de maven.
- Cancelado el trabajo muestra el progreso

Visualización de datos

Algunos gráficos básicos ya están incluidos en Apache Zeppelin. Las visualizaciones no están limitadas a las consultas con Spark SQL, sino que cualquier salida desde cualquier tipo de lenguaje puede ser reconocida y visualizada.

Gráfico dinámico

Apache Zeppelin agrega valores y los muestra en el gráfico dinámico con solo arrastrar y soltar. Se puede crear fácilmente un gráfico con múltiples valores agregados que incluyen suma, recuento, promedio, mínimo, máximo.

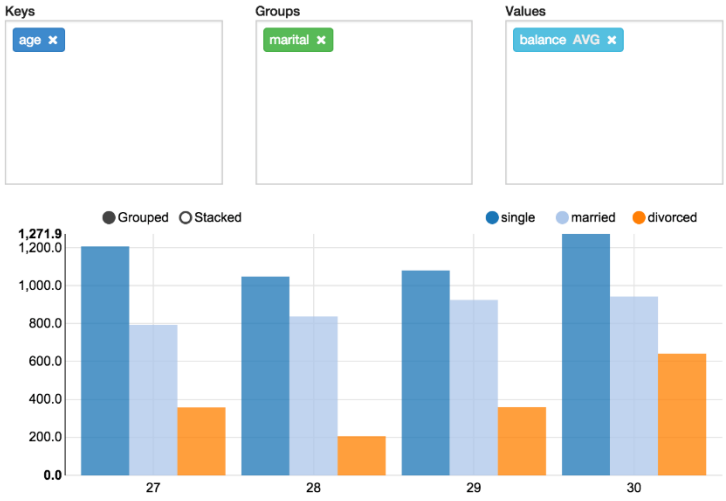


Ilustración 14 - Muestra de output de consulta

Ventajas

Apache Zeppelin ofrece diversas ventajas como son:

- **Máxima Simplicidad**: Permite, de una forma muy intuitiva, manejar conexiones de datos, rápidas, con fuentes de datos en Hive, HBae, etc… o SQL más tradicional como PostgreSQL o MySQL. Para entonces construir aplicaciones muy rápidamente, visualmente y que poder compartirlas rápidamente.

- **Agnóstico del lenguaje**: Gracias a una gran disponibilidad de plugins o interpreter, se podrán hacer múltiples acoplaciones que van a hacer más grande nuestra solución.

- **Interfaz sobre Bootstrap y AngularJS**: Visualmente es muy agradable y tiene una facilidad para poder adecuarlo todavía más a nuestras preferencias, aunque, los usuarios que lo visualicen también podrán hacer los cambios que requieran para adecuarlo todavía más.

- **Compartir Notes:** Notebook permite compartir tus procesos con otros, de modo que estos puedan ser entendidos, modificados y/o adaptados según las necesidades. Además permite exportar las notes que desees y ejecutarlo en otra instancia de Zeppelin.

CASO PRÁCTICO

API DE TWITTER

Una API REST es un servicio que provee de funciones que ofrecen la posibilidad de hacer uso de un servicio web que no es nuestro, dentro de nuestra aplicación de forma segura. Al usar una API todo el desarrollo que se quiera realizar está limitado por los métodos o funciones que la aplicación incluya, en otras palabras, no se pueden añadir funcionalidades nuevas. Es una forma de que las compañías se aseguren qué puedes y no pueden hacer los clientes desarrollados por terceros.

En nuestro caso la API de Twitter la vamos a utilizar únicamente para leer datos de Twitter. La API pública de Twitter proporciona acceso a un alto volumen de tweets con una baja latencia. La API de Twitter no se puede utilizar de forma directa, sino que se tiene que crear una aplicación desde la que podremos autentificar y realizar acciones con la API.

Crear una cuenta de desarrollador

El primer paso es crear una cuenta de desarrollador. Para ello se debe de acceder a la página de desarrolladores de Twitter:

https://developer.twitter.com/

Dentro de la página lo único que se debe de hacer es identificarse con nuestra cuenta de Twitter habitual.

Crear una aplicación

Una vez que se tiene una cuenta de desarrollador. El siguiente paso es crear una aplicación para ello hay que dirigirse a la página de gestión de aplicaciones:

https://apps.twitter.com/

Se hace click en el botón “Create New App” y se rellenan los campos que se solicitan con los datos que de nuestra APP. Solicitan un sitio web que no se va a necesitar, por lo que se añade una Url válida para que permita crear nuestra aplicación.

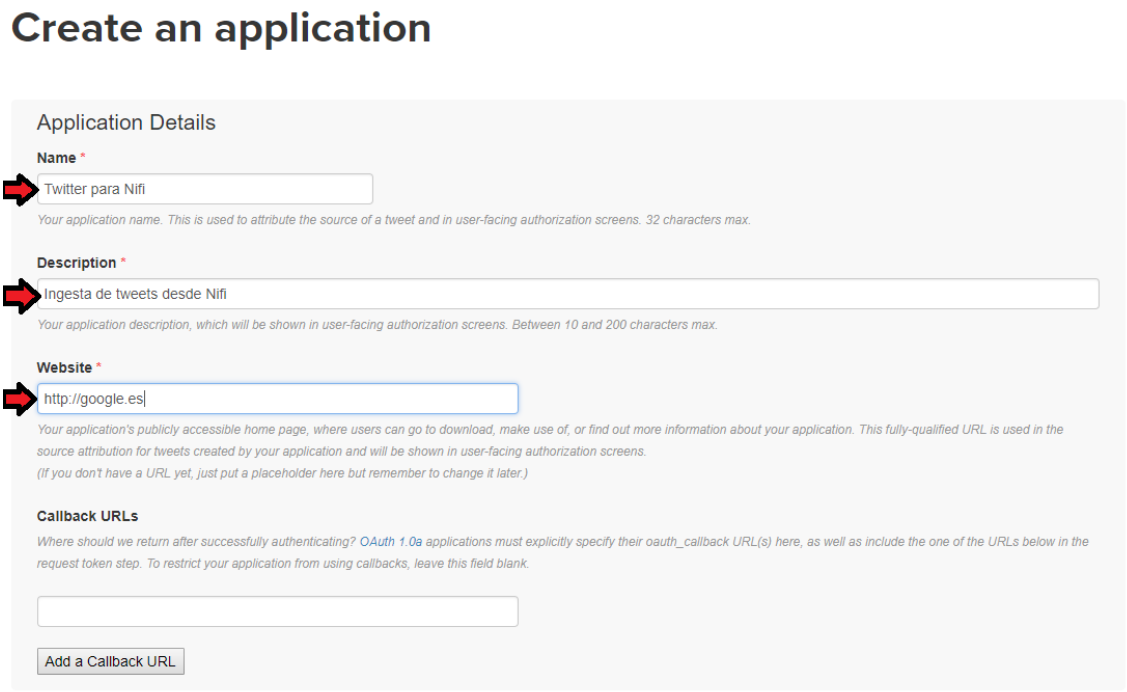


Ilustración 15 - Crear aplicación Twitter

A continuación se acepta el acuerdo de desarrollador de Twitter y pulsamos en “Create your Twitter application”

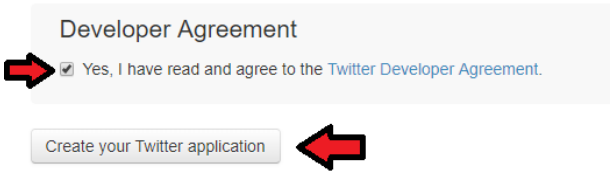


Ilustración 16 - Crear aplicación Twitter

Obtención de las claves de acceso

Para la comunicación con la API de Twitter se necesitará conocer las diferiremos claves asociadas a nuestra aplicación. Estas claves son:

- Consumer Key (API Key)
- Consumer Secret (API Secret)
- Acces Token
- Acces Token Secret

Para ello hay que dirigirse de nuevo a la página de gestión de aplicaciones y seleccionar la aplicación que se acaba de crear.

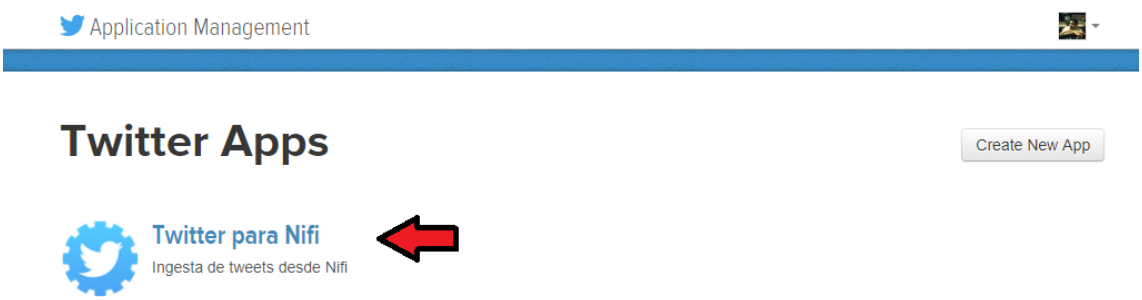


Ilustración 17 - Obtención de las claves de acceso

Hay que dirigirse a la pestaña Keys and Acces Tokens. Al haber creado la aplicación, Twitter ha generado automáticamente el Consumer Key y el Consumer Secret.

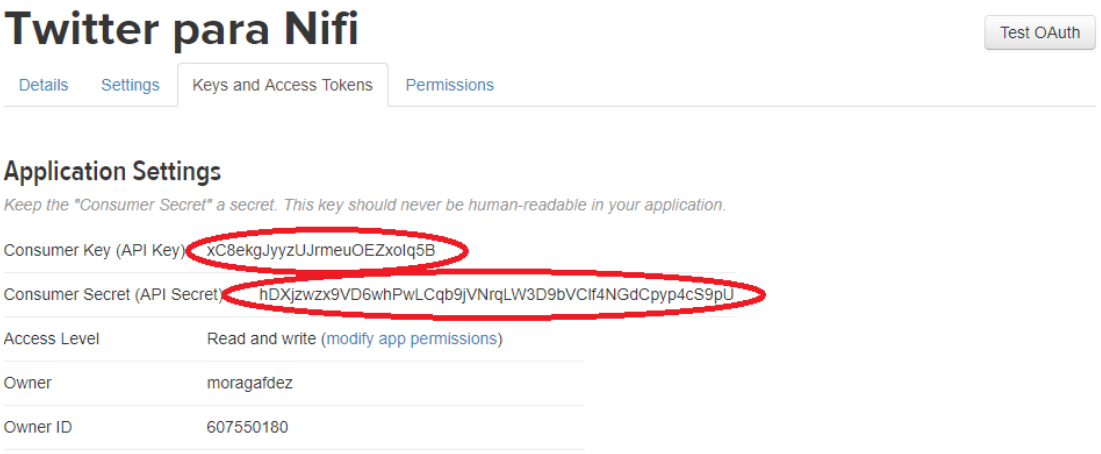


Ilustración 18 - Obtención de las claves de acceso

Sin embargo para autorizar las llamadas a la API, se debe crear los Token. Para ello, se pincha en “Create my Access token”. Los tokens se generarán con el nivel de permiso que tenga la aplicación.

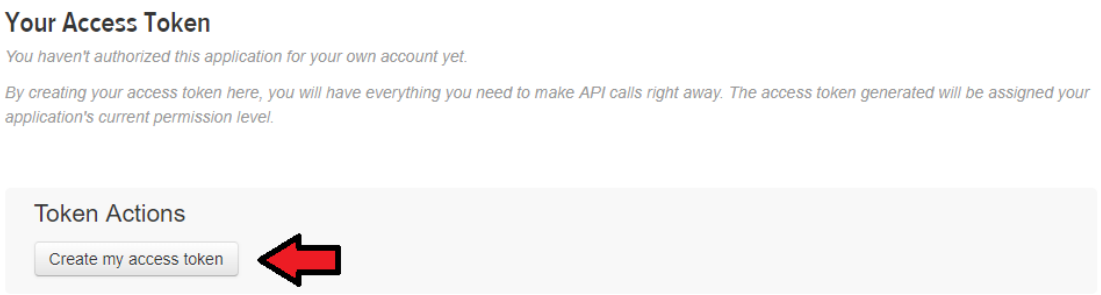


Ilustración 19 - Obtención de las claves de acceso

Una vez creados los tokens, ya se puede consultar nuestro Access Token y Access Token Secret y tener todos los datos disponibles que harán falta para las futuras conexiones a la API de Twitter.

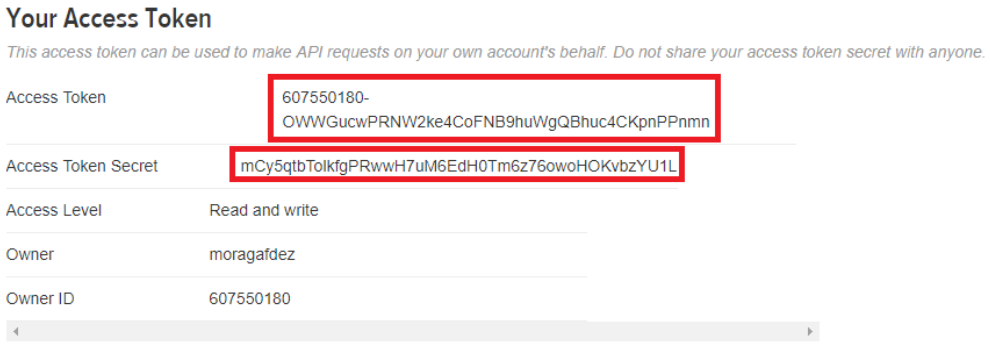


Ilustración 20 - Obtención de las claves de acceso

Cambio de permisos de la aplicación

Por lo requisitos que va a tener nuestro sistema, a la API de Twitter sólo se le va a hacer solicitudes de datos, es decir, nosotros desde nuestra aplicación no se va a hacer acciones de escritura ni de acceder a mensajes directos, que son los otros dos niveles de permiso que existen. Por lo tanto a nuestra aplicación hay que dar el nivel de permiso “Read Only”.

Para ello hay que dirigirse a la pestaña “Permissions”, seleccionar el nivel de permiso requerido y pinchar en “Update Settings”

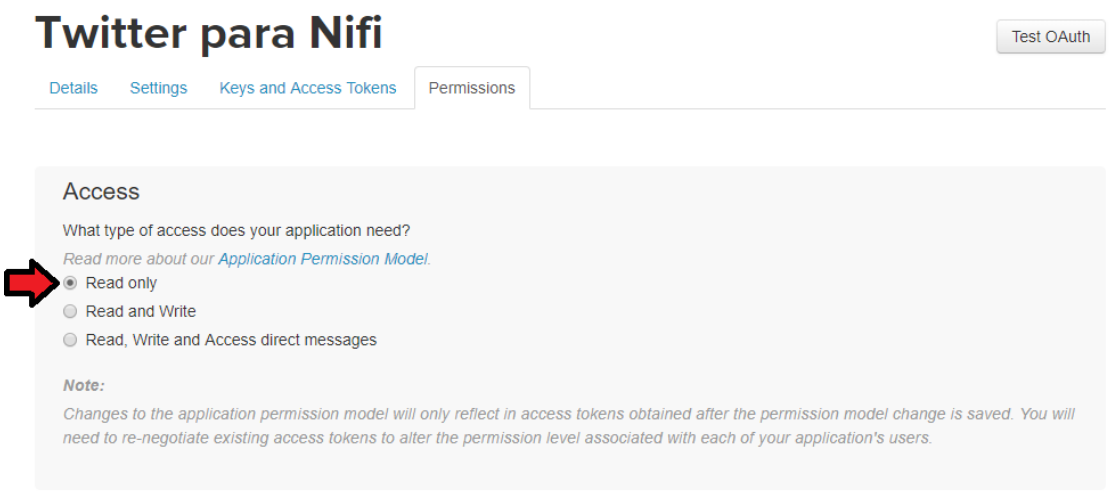


Ilustración 21 - Cambio de permisos de la aplicación

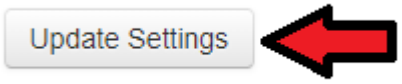


Ilustración 22 - Cambio de permisos de la aplicación

Regeneración de claves de acceso

Como bien nos indican en la note, una vez que se ha hecho un cambio de permisos, se necesita volver a generar las cuatro claves de la aplicación, por lo tanto se va a la pestaña “Keys and Access Tokens” y se regenera la Consumer Key y Secret, así como, el Acces Token y Access Token Secret.

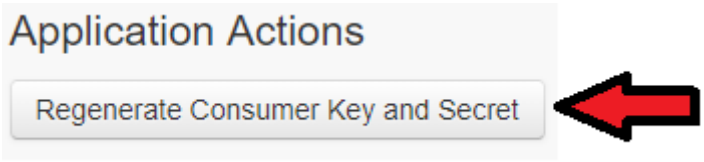


Ilustración 23 - Regeneración de claves de acceso

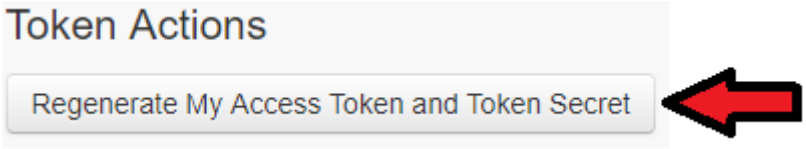


Ilustración 24 - Regeneración de claves de acceso

Ahora sí, después de haber seguido todos estos pasos ya se disponen de nuestras claves de acceso definitivas y con los el nivel de permiso necesario para poder acceder a la API de Twitter.

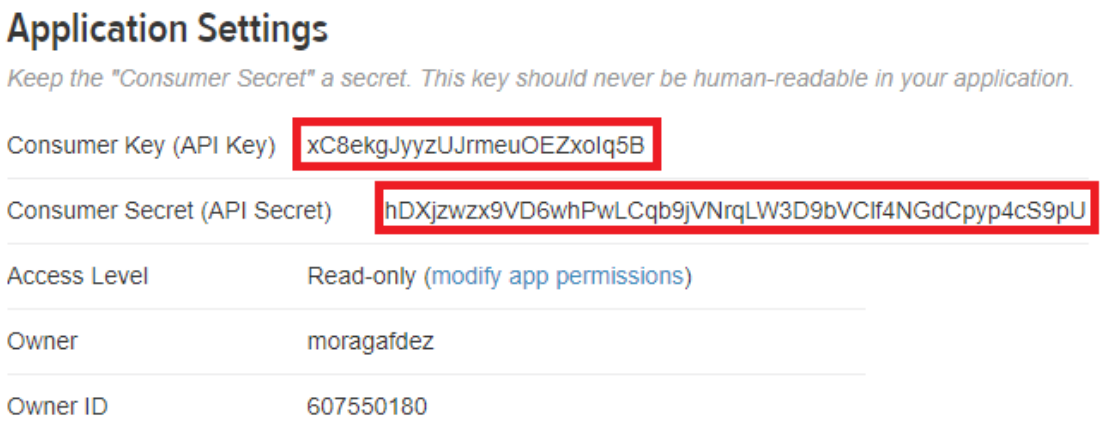


Ilustración 25 - Regeneración de claves de acceso

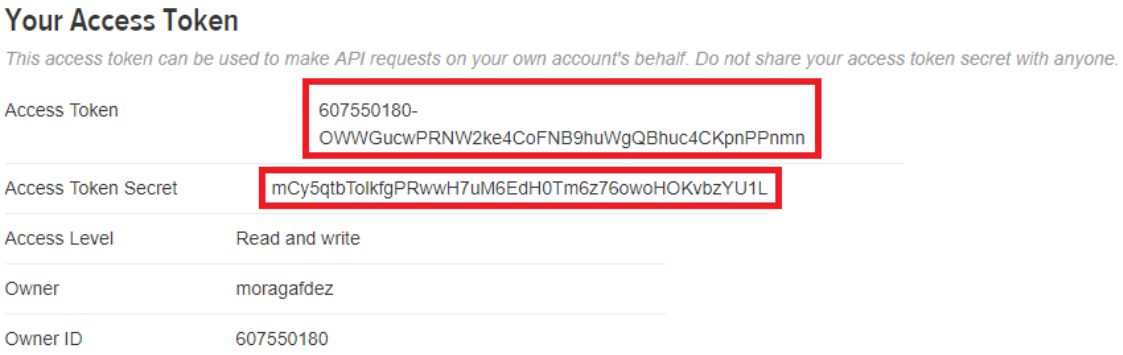


Ilustración 26 - Regeneración de claves de acceso

INGESTA DE DATOS CON APACHE NIFI

Instalación de la herramienta

Se van a explicar los pasos necesarios para la instalación de Apache Nifi en nuestro ordenador. El primer paso es descargar el programa. Se dispone de la versión Apache Nifi 1.6.0. Para descargarse esta herramienta hay que dirigirse a la pestaña de descargas de la web de Apache Nifi:

https://nifi.apache.org/download.html

Una vez aquí, se selecciona el tipo de archivo y la versión deseada. Por consiguiente, se seleccionan los binarios de la versión 1.6.0.

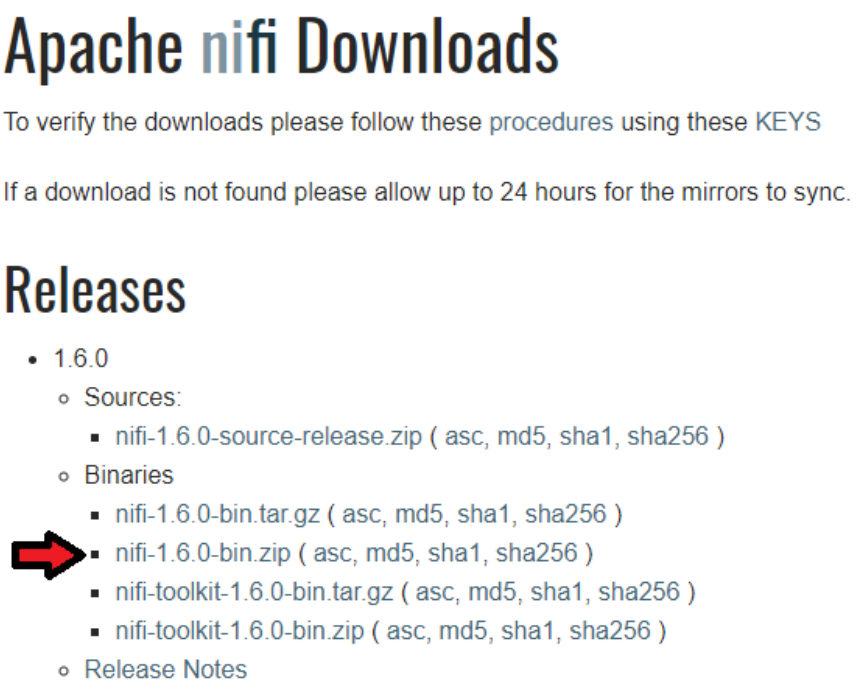


Ilustración 27 - Instalación de la herramienta

Se redirige a la web de Apache Software Foundation. Aquí se pincha en el enlace de descarga recomendado y acto seguido comienza la descarga de la herramienta.

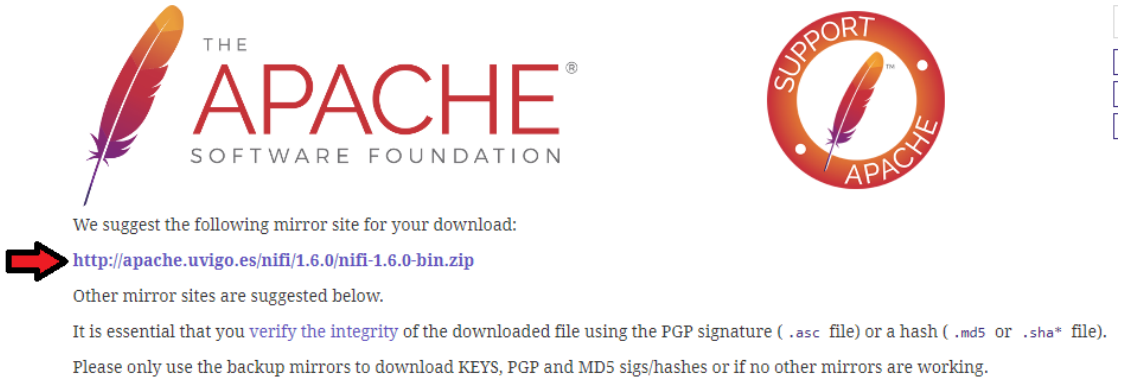


Ilustración 28 - Instalación de la herramienta

Una vez que ya se tiene descargado el archivo comprimido, se deb descomprimirlo. En nuestro caso, para acceder en el futuro a la herramienta lo más rápido posible, se ha decidido descomprimirlo en el directorio C:\

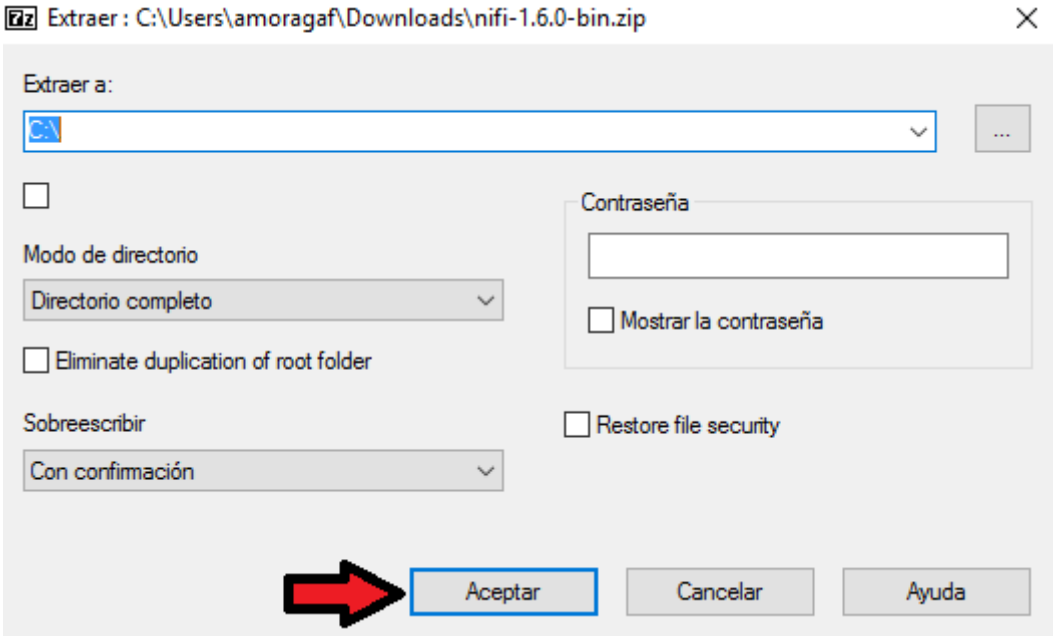


Ilustración 29 - Instalación de la herramienta

Una vez hecho esto ya se tendrían todos los archivos necesarios para posteriormente levantar el servicio de Apache Nifi.

Cambiar el puerto de apache nifi

Como ya hemos comentado anteriormente cuando hemos hablado más en detalle de la herramienta en sí, la forma en la que se va a trabajar con ella es a través de la interfaz web. La dirección que por defecto se debe de introducir en nuestro navegador para acceder a la interfaz es http://<hostname>:8080/nifi. Se observa que por defecto el puerto en el que se levanta el servicio el puerto 8080. Si sólo utilizásemos Apache Nifi, no supondría ningún problema y se podría dejar la configuración que trae por defecto Nifi. Sin embargo en nuestro entorno Big Data hay un conflicto entre dos herramientas ya que Apache Nifi y Apache Zeppelin trabajan en el mismo puerto, es decir ambas herramientas utilizan el puerto 8080.

Para solucionarlo se ha optado por modificar el puerto en el que se va a levantar Apache Nifi. Para ello hay que dirigirse al archivo "nifi.properties" que se encuentra en la ruta:

C:\nifi-1.6.0\conf\nifi.properties

Se abre el archivo con un editor de texto con el que se modificará el parámetro que se desea, en nuestro caso se utiliza el programa Notepad.

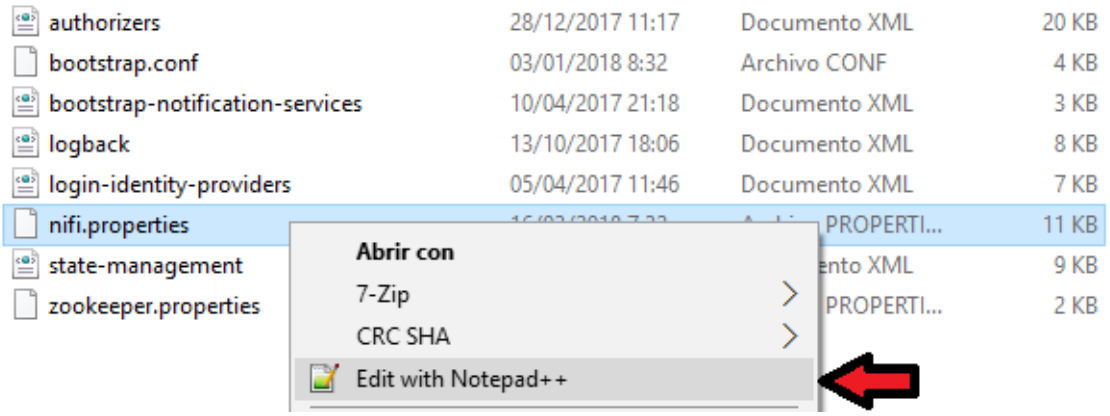


Ilustración 30 - Cambiar el puerto de apache nifi

Ahora únicamente lo que hay que hacer es localizar el parámetro al que corresponde al puerto que utiliza Nifi. Ese parámetro lo denomina “nifi.web.http.port”. El nuevo puerto que le se asignará es el 8081.

```
# web properties #
nifi.web.war.directory=./lib
nifi.web.http.host=
nifi.web.http.port=8081
nifi.web.http.network.interface.default=
nifi.web.https.host=
nifi.web.https.port=
nifi.web.https.network.interface.default=
nifi.web.jetty.working.directory=./work/jetty
nifi.web.jetty.threads=200
nifi.web.max.header.size=16 KB
nifi.web.proxy.context.path=
nifi.web.proxy.host=
```

Ilustración 31 - Cambiar el puerto de apache nifi

Una vez cambiado el puerto, la única diferencia para nosotros como usuarios es que a la hora de arrancar la interfaz web se deberá introducir la dirección localhost:8081/nifi

Inicializar la herramienta

Para arrancar Nifi en nuestro ordenador hay que dirigirse al directorio donde se ha descomprimido el archivo que hemos descargado que contenía la herramienta. Una vez ahí se pincha en la carpeta “bin” y se hace doble click en el archivo bat llamado “run-nifi”.

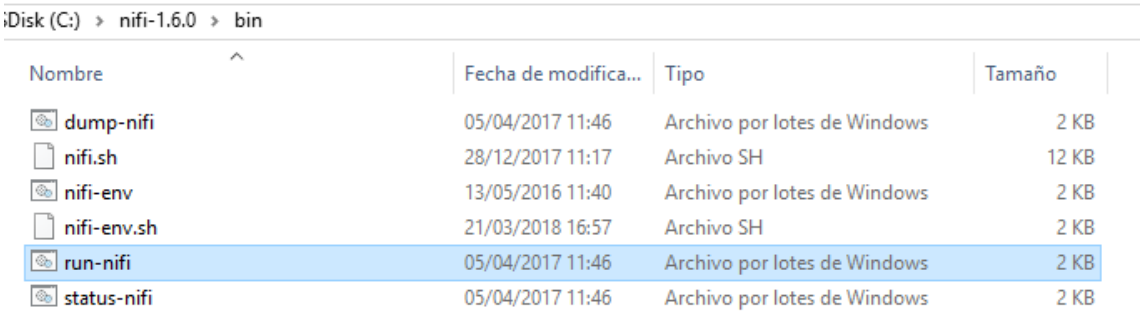


Ilustración 32 - Inicializar la herramienta

Cuando comience a ejecutarse aparecerá una ventana de consola indicando que se está inicializando el servicio.

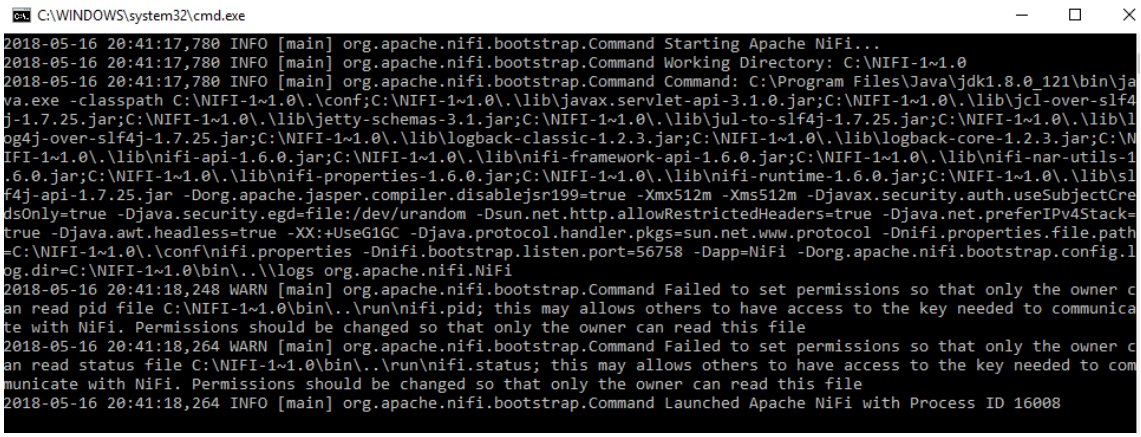


Ilustración 33 - Inicializar la herramienta

Pasados unos instantes ya se ha inicializado Apache Nifi y se puede empezar a trabajar con él a través de su interfaz web. Para ello se pone la dirección http://localhost:8081/nifi/ en nuestro navegador web. Se observa que se ha iniciado correctamente.

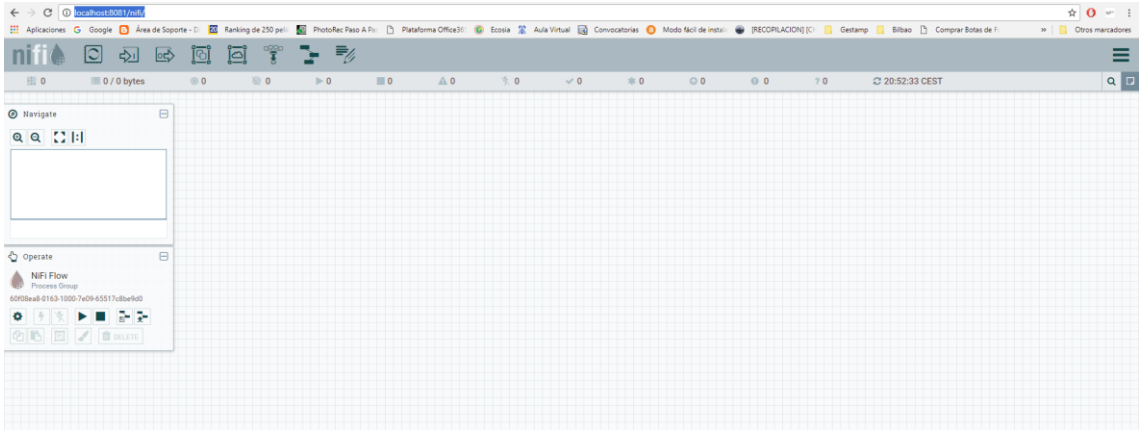


Ilustración 34 - Inicializar la herramienta

Creación del flujo. De Twitter a Kafka

Una vez se ha conseguido la correcta instalación y configuración de Nifi, se va a crear un flujo de trabajo que permitirá establecer la conexión con la API de Twitter para hacer la ingesta de datos. Una vez que se tienen los datos en el flujo de Nifi, se debe conseguir que hacia Kafka vayan los datos como se desea. Para ello, se va a realizar un primer filtro con los tweets que no estén vacíos y que además estén en español. Por último y la que es la finalidad de este flujo, se llevarán los datos deseados a un topic de Kafka.

La creación del flujo de trabajo en Nifi se realiza a través de la conexión de los diferentes processors. Para añadir un processor se pincha sobre el icono del processor que es el izquierdo de la Components Toolbar (ver Ilustración 6) y se arrastra hacia la cuadrícula principal. Aparecerán todos los processors existentes. El primero de los processor que se necesitará es el GetTwitter que se encargará de la conectar con la API de Twitter y capturar los datos.

Para ello se pone en el buscador el nombre del processor, se selecciona y se pulsa en añadir.

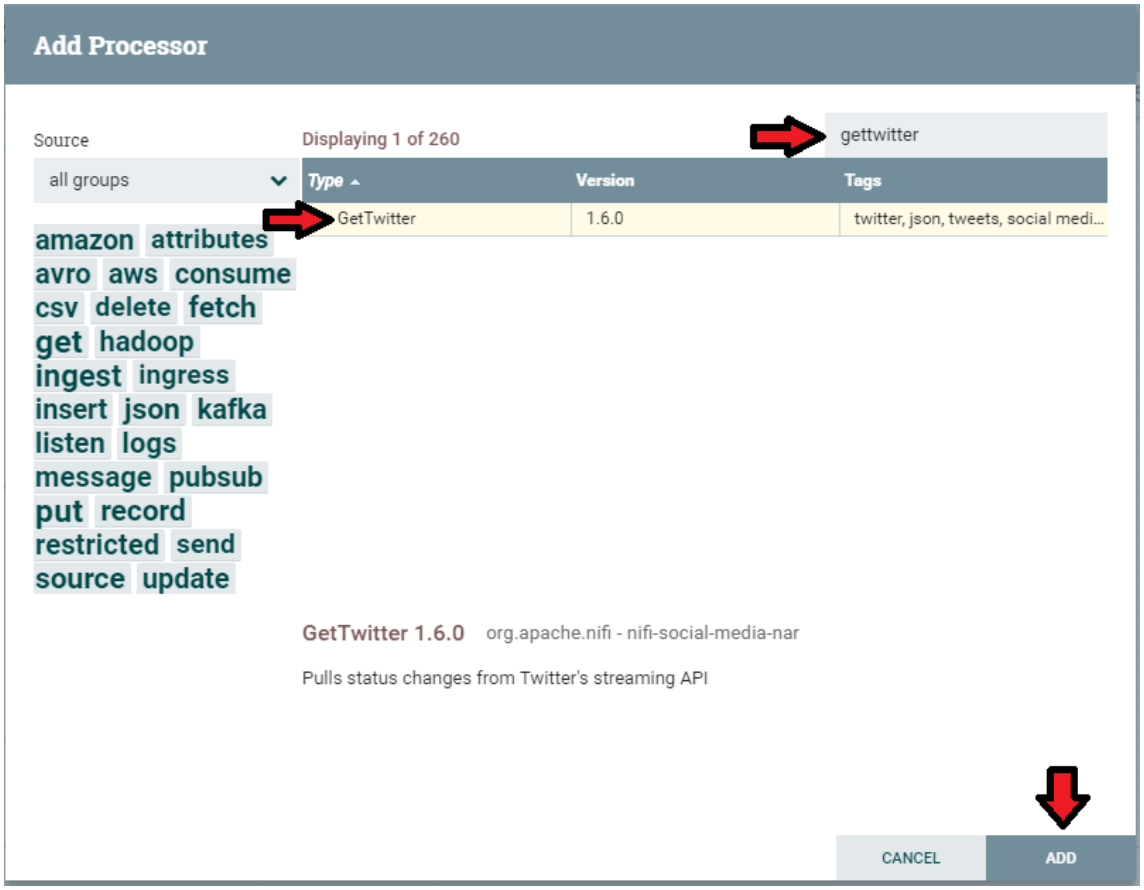


Ilustración 35 - Creación del flujo. De Twitter a Kafka

Una vez añadido, el processor se debe configurar. Para ello se hace doble click sobre el processor y hay que dirigirse a la pestaña “Properties”. Aquí aparecen los distintos parámetros de configuración que tiene este processor.

Se configuran las claves de acceso a la API de Twitter que se han obtenido anteriormente y como no se va a aplicar ningún tipo de filtro en este processor, el parámetro “Twitter Endpoint” hay que dejarlo con su valor por defecto es decir “Sample Endpoint”.

Una vez configurado se pulsa el botón “Apply”.

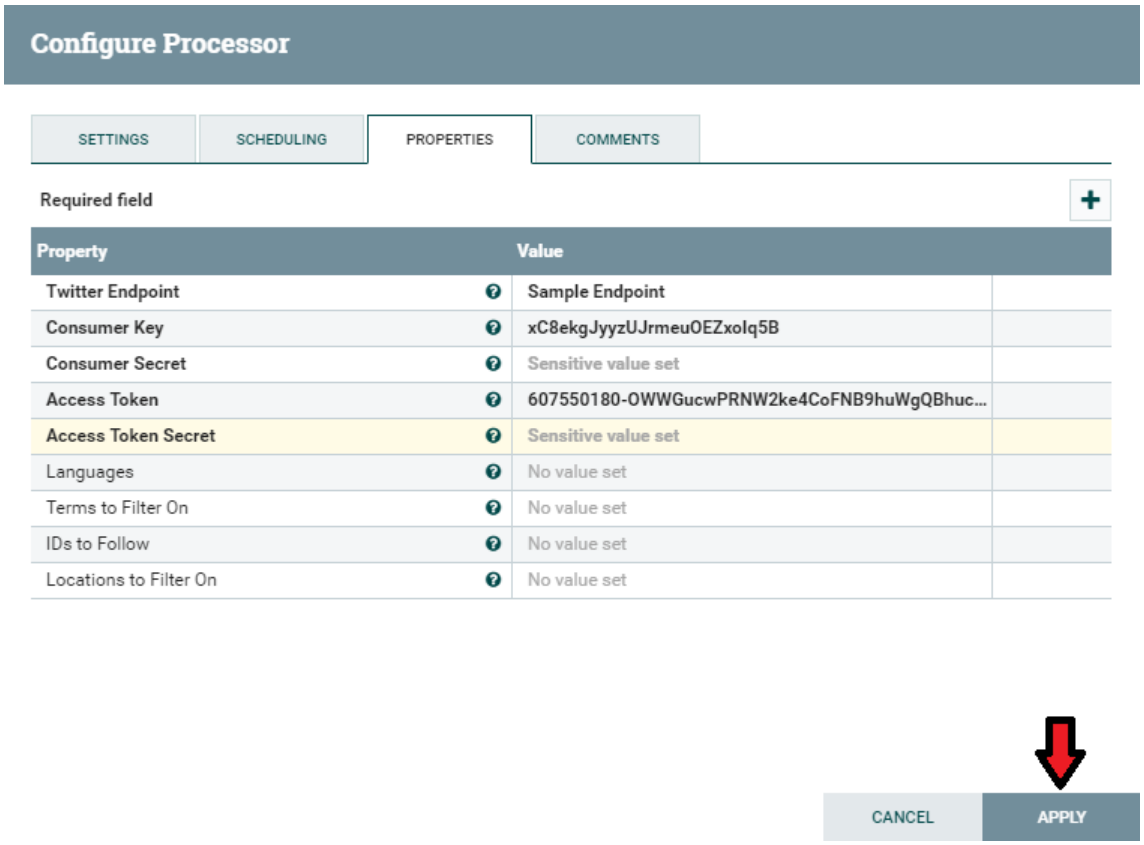


Ilustración 36 - Creación del flujo. De Twitter a Kafka

Los parámetros que se han modificado han sido:

- **Consumer Key:** xC8ekgJyyzUJrmeuOEZxoIq5B
- **Consumer Secret:** hDXjzwzx9VD6whPwLCqb9jVNrqLW3D9bVClf4NGdCpyp4cS9pU
- **AccesToken:** 607550180-OWWGucwPRNW2ke4CoFNB9huWgQBhuc4CKpnPPnmn
- **Acces Token Secret:** mCy5qtbTolkfgPRwwH7uM6EdH0Tm6z76owoHOKvbzYU1L

Como se puede observar, este processor tiene parámetros para realizar varios filtros a los tweets que ingesta, pero para demostrar el potencial de Apache Nifi se ha decido realizar ese filtro con diferentes processors.

El primero de estos processors necesarios será el EvaluaJsonPath. Este processor se utiliza para seleccionar los atributos del Json por los que se realizará el posterior filtro. Hay que aclarar que el formato en el que la API de Twitter proporciona los tweets es en formato Json. Se añade el processor arrastrando el icono correspondiente.

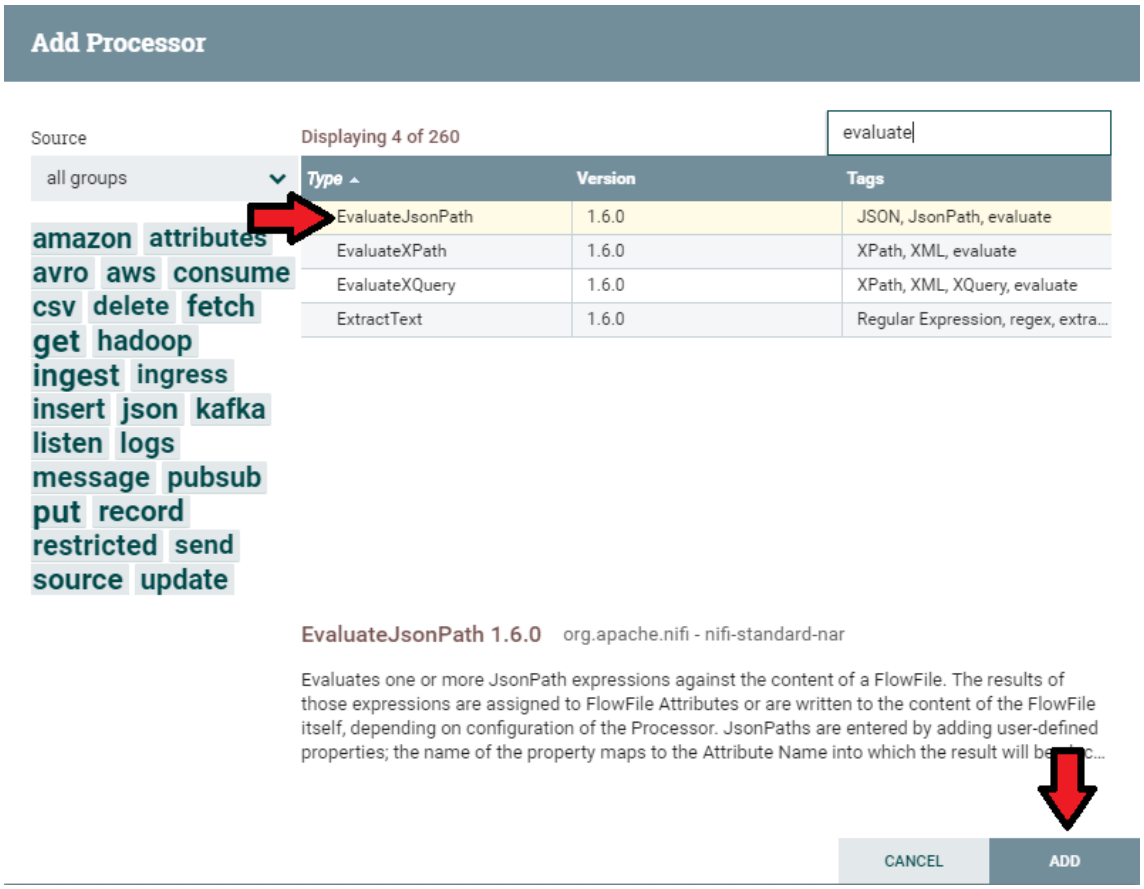


Ilustración 37 - Creación del flujo. De Twitter a Kafka

Una vez añadido se hace doble click sobre él y se accede a sus propiedades. En el campo "Destination" se selecciona en el desplegable la opción "flowfile-attribute". Para seleccionar los campos del Json por los que posteriomente se realizará el filtrado, hay que crearse dos parámetros de configuración. Para ellos se hace click en el botón "+". Se le asigna de nombre a la propiedad "twitter.lang" y le da de valor "$.lang". Con esto se está haciendo que esta propiedad haga referencia al atributo que indica el lenguaje del tweet.

Para el segundo de los parámetros que se tiene que crear se sigue el mismo procedimiento. Como nombre del parámetro le asignamos “twitter.text” dándole de valor “$.text”. Hará referencia al campo texto del tweet. Una vez finalizada la configuración del processor se pulsa el botón “Apply”

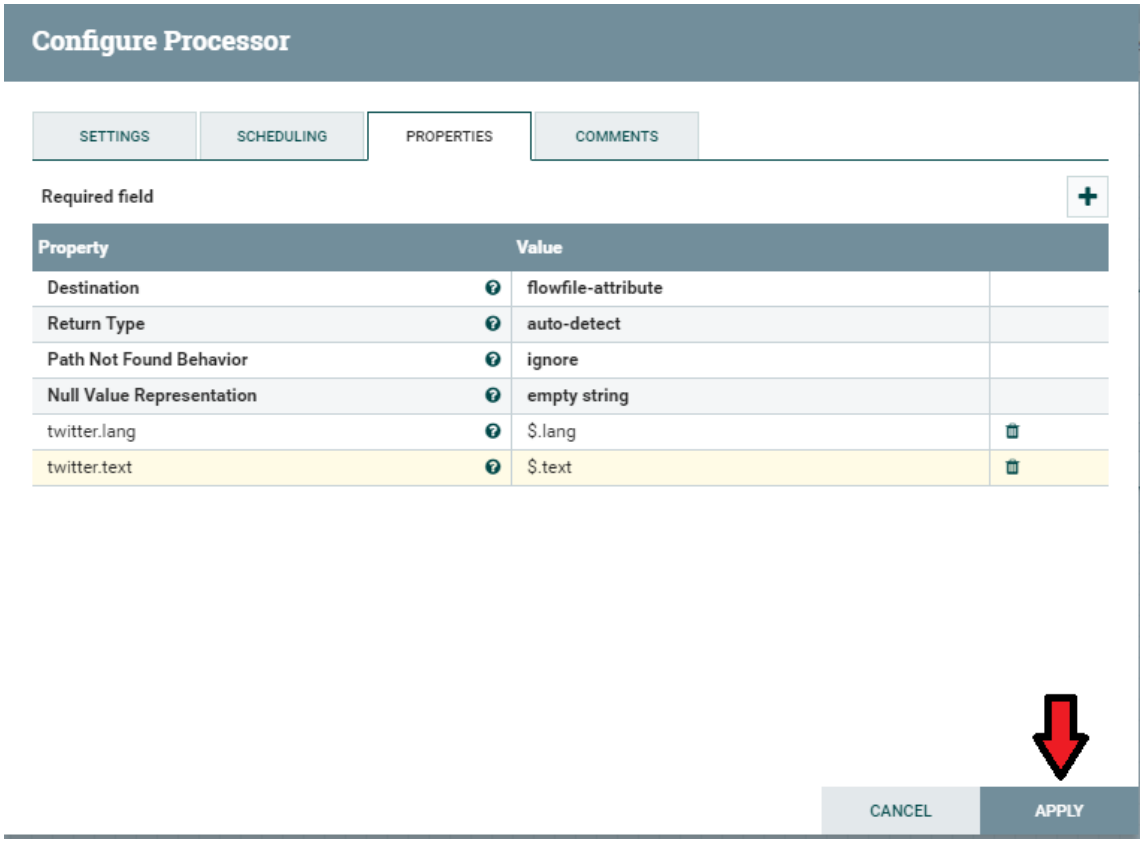


Ilustración 38 - Creación del flujo. De Twitter a Kafka

Los parámetros que se han modificado han sido:

- **Destinarion:** flowfile-attribute
- **Twitter.lang:** $.lang
- **Twitter.text:** $.text

Una vez se tienen estos dos processors creados con sus correspondientes parámetros de configuración, el siguiente paso que a realizar es conectar ambos processors para que el flujo de flowfiles vaya de un processor a otro. Para ello, se selecciona el primero de los processor, es decir el GetTwitter, y aparece un símbolo de conexión. Lo único que hay que hacer es pinchar sobre él y unirlo con el processor de EvaluateJsonPath.

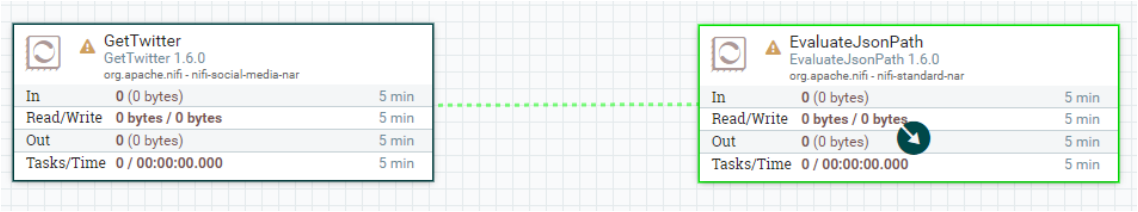


Ilustración 39 - Creación del flujo. De Twitter a Kafka

Una vez que se ha conectado, aparece el menú de creación de conexión. Únicamente aseguramos que el flujo se va a realizar con los flowfiles que se creen con éxito. Para ello en la opción de "For Relationships" debe estar seleccionada únicamente el valor "success". Las distintas configuraciones del conector las dejamos con su valor por defecto. Una vez comprobado todo ello se pulsa el botón "Add".

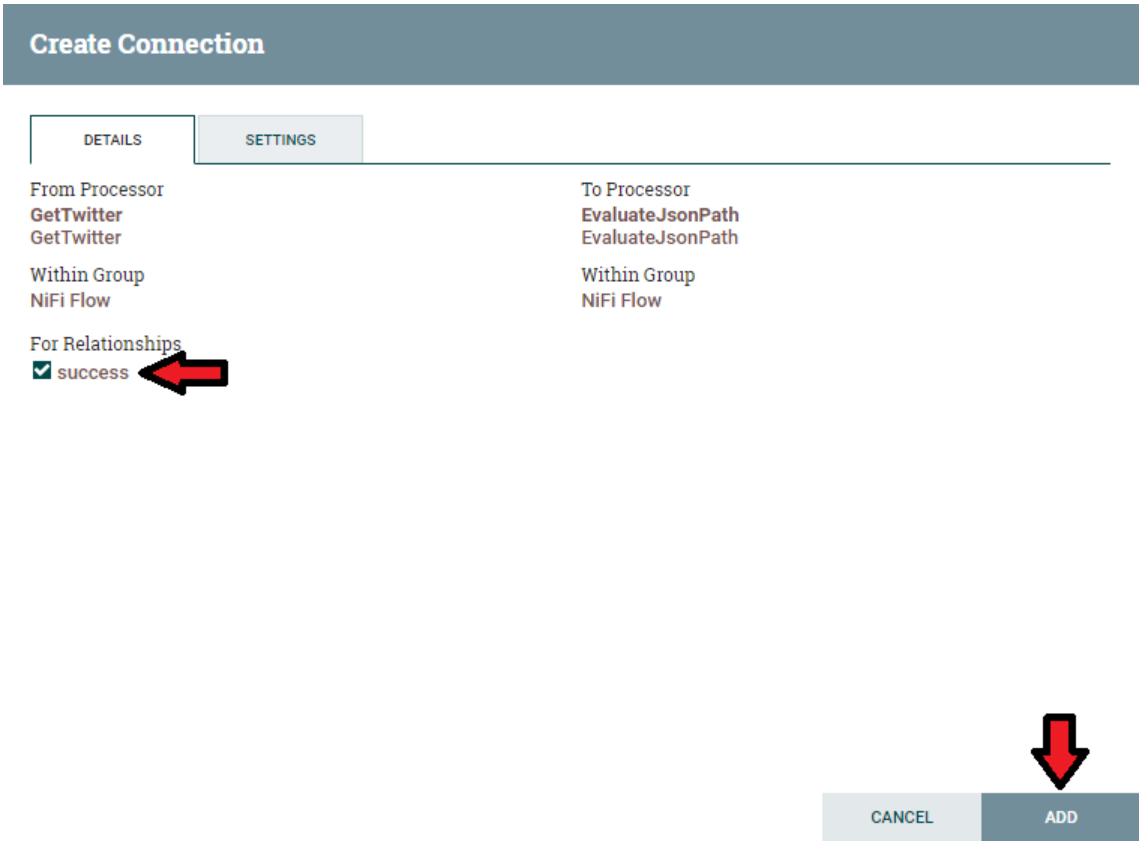


Ilustración 40 - Creación del flujo. De Twitter a Kafka

El siguiente paso que se va a realizar es la configuración de un processor que realice el filtrado de los tweets por una serie de condiciones sobre los atributos que se han indicado en el "EvaluateJsonPath". Ese requisito se va a realizar con el processor llamado "RouteOnAtributte". Para añadirlo se seguirá el mismo proceso que se ha realizado para los anteriores processor.

Se arrastra el icono del processor hasta la cuadrícula de trabajo, se filtra por ejemplo por "route", se selecciona el processor "RouteOnAtributte" y se hace click en el botón "Add".

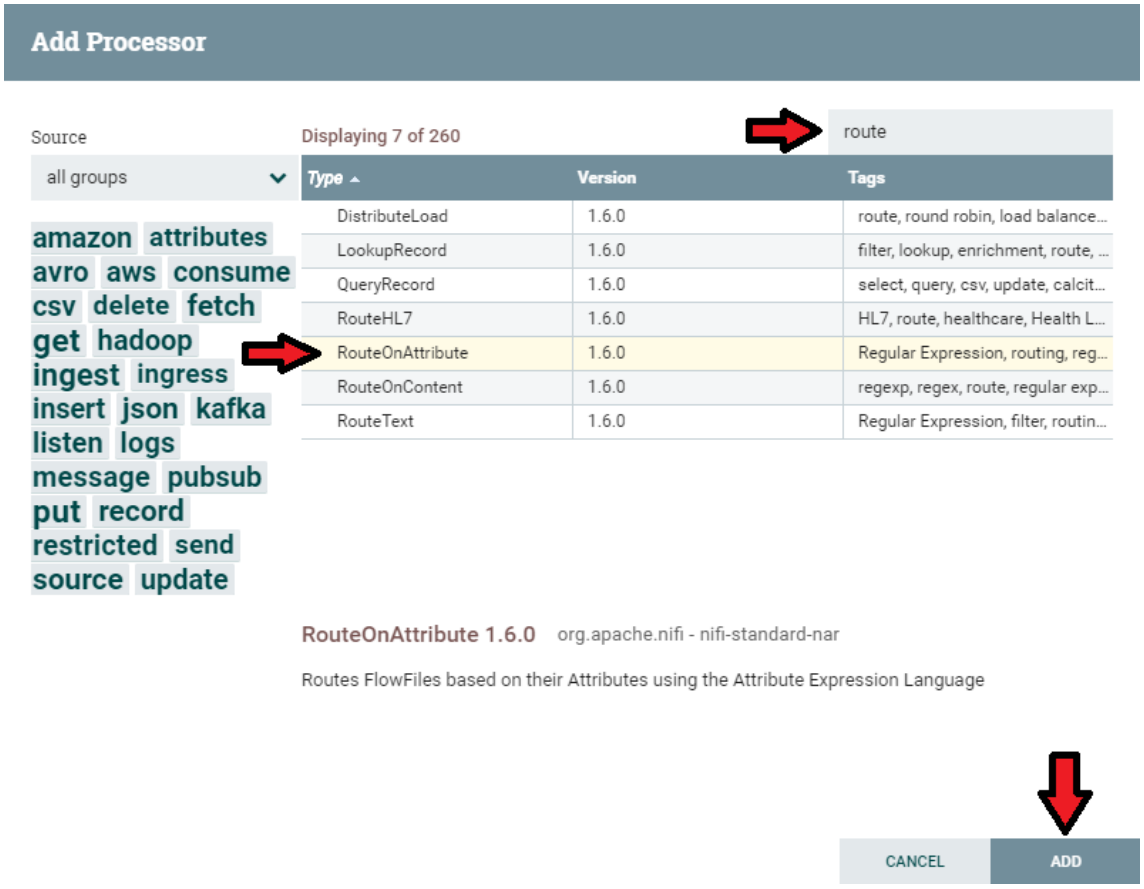


Ilustración 41 - Creación del flujo. De Twitter a Kafka

Se acceden a sus propiedades haciendo doble click sobre el processor que se acaban de añadir y pinchando en la pestaña "Properties". La propiedad de configuración "Routing Strategy" se debe dejar su valor por defecto "Route to Property name". Para que el processor haga el filtrado de los tweets, hay que crear una nueva propiedad de configuración con una condición que si se cumple devolverá valor "true" y se enrutarán los tweets hacia el siguiente processor.

Para añadir esta nueva propiedad de configuración, se hace click en el botón “+”, dándole de nombre de la propiedad “tweets” y como valor de la condición "${twitter.text:isEmpty():not():and(${twitter.lang:equals("es")})}".

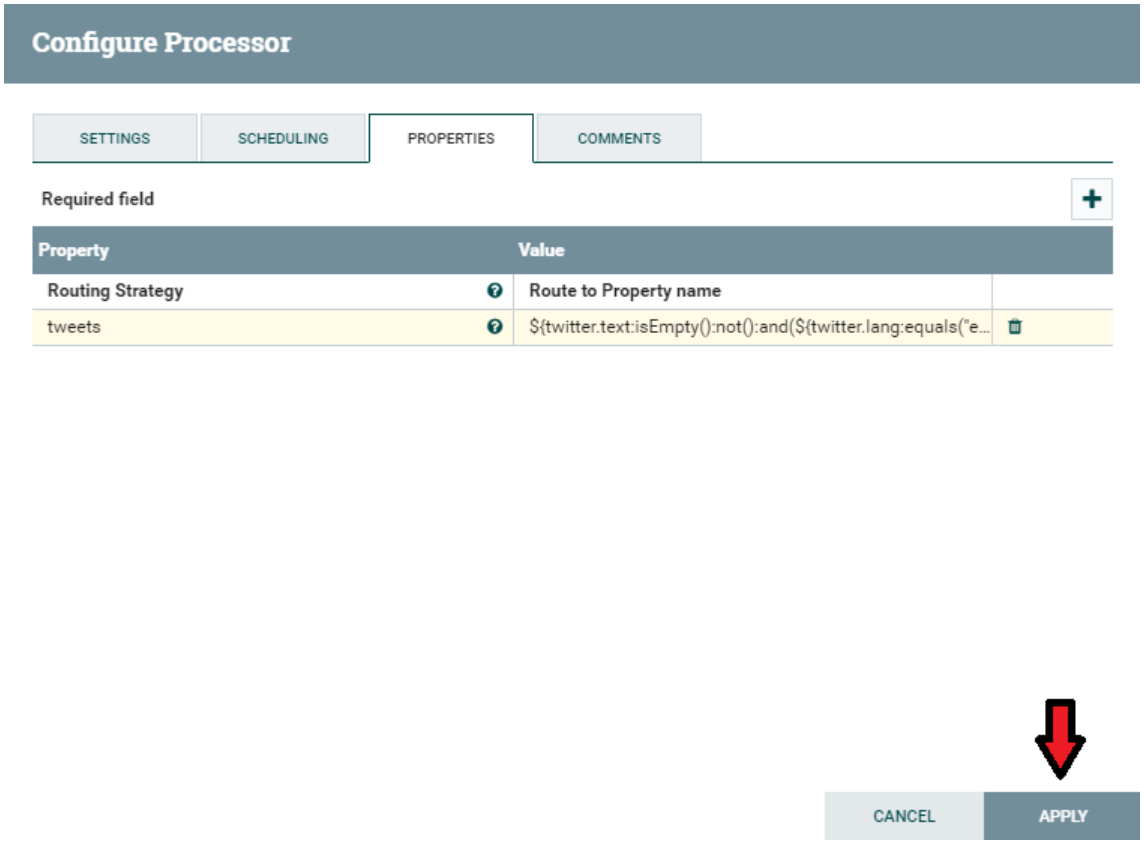


Ilustración 42 - Creación del flujo. De Twitter a Kafka

Si la condición “tweets” devuelve valor true significará que los tweets son en español y no están vacíos.

Los parámetros de configuración que se han modificado han sido:

- **Tweets:** ${twitter.text:isEmpty():not():and(${twitter.lang:equals("es")})}

Una vez se han configurado completamente el “RouteOnAttribute”, es el momento de conectarlo con el processor de donde le deben provenir los flowfiles, es decir, el “EvaluateJsonPath”.

Para realizar la conexión se selecciona el processor “EvaluateJsonPath”, se pincha sobre el símbolo de conexión y se arrastra hasta el “RouteOnAttribute”.

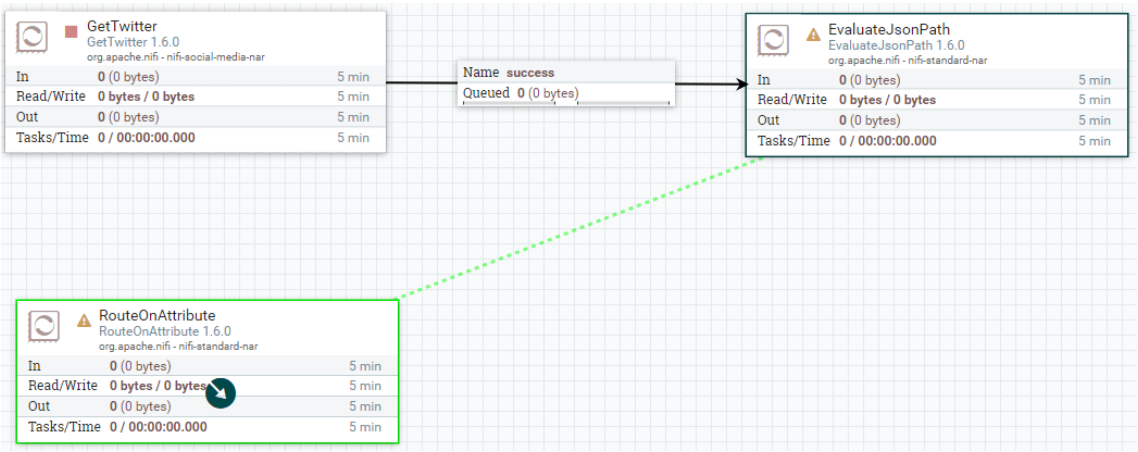


Ilustración 43 - Creación del flujo. De Twitter a Kafka

Una vez hecha la conexión, Nifi mostrará el menú de creación de la conexión. De las opciones de relación se selecciona únicamente “matched”, se deja la configuración por defecto y se pulsa en “Add”.

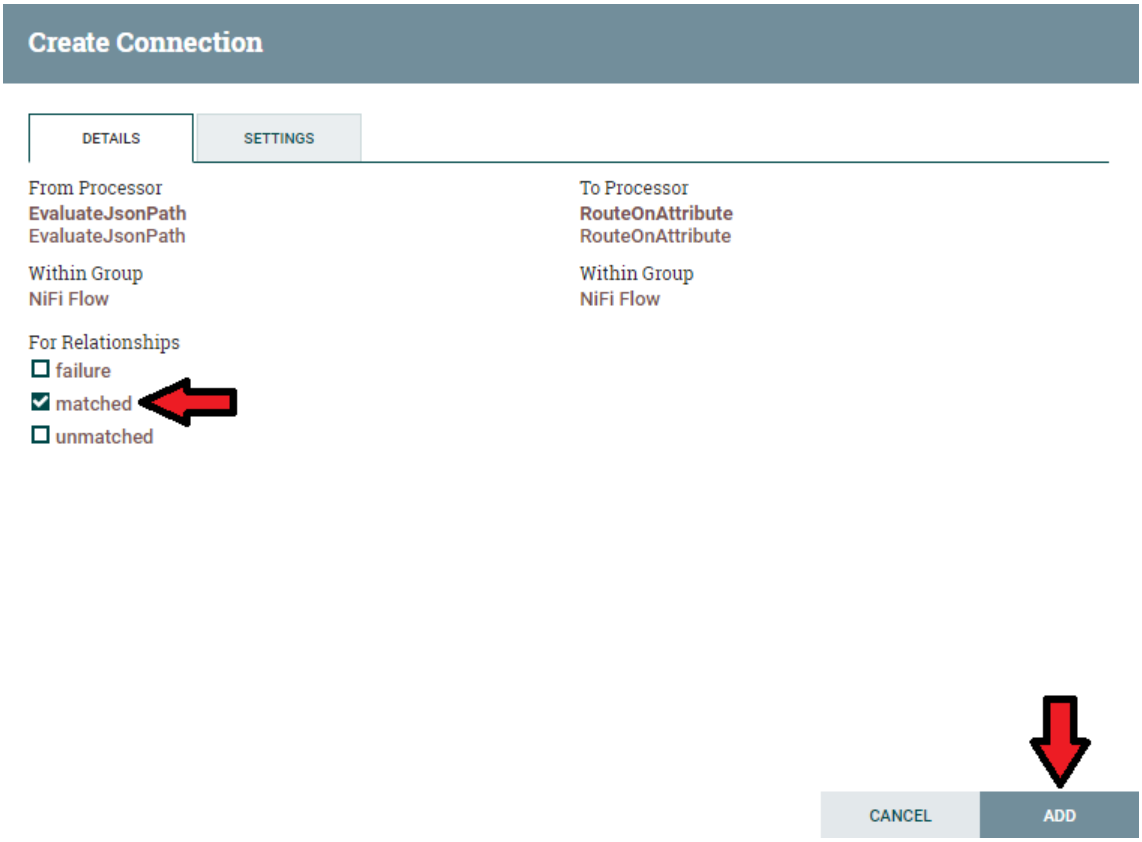


Ilustración 44 - Creación del flujo. De Twitter a Kafka

Llegados a este punto, ya se ha realizado el flujo de trabajo para conectarse a la API de Twitter, hacer la ingesta de datos y un primer filtro del dataset deseado para su posterior almacenamiento y procesamiento. El último paso que hay que realizar en Nifi, es el envío de todos los tweets que se están ingestando y filtrando al topic de Kakfa deseado. Para acometer esta necesidad se va a utilizar el processor “PutKafka”. Como se ha ido haciendo con los demás processor, para añadirlo se arrastra el icono correspondiente hacia la cuadrícula de trabajo, se filtra por ejemplo por “put”, s slecciona el processor “PutKafka” y se hace click en el botón “Add”.

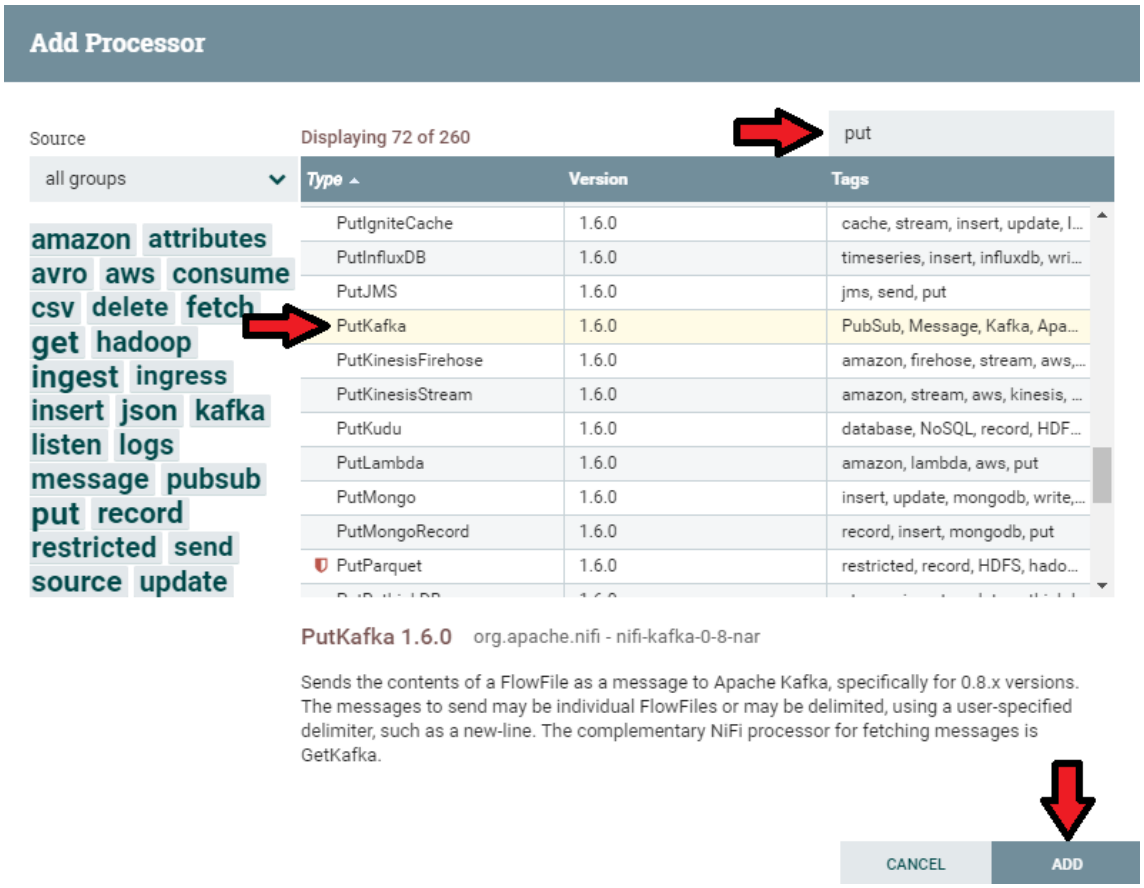


Ilustración 45 - Creación del flujo. De Twitter a Kafka

Una vez añdido, se hace doble click sobre el processor para acceder a su configuración y hay que dirigirse a la pestaña de “Properties”. De las diferentes propiedades, hay que configurar los campos Known Brokers, Topic Name y Client Name de acuerdo a nuestro entorno de Kafka desplegado. En la propiedad de configuración “Known Brokers” tenemos que indicar el nodo de Kafka en formato <host>:<port>. La dirección del nodo en el que se va a levantar Kafka y por lo tanto hay que poner de valor a la propiedad “Known Brokers” es “localhost:9092”.

Para indicar en qué topic deberá almacenar los datos, aparece el campo “Topic Name” al que hay que asignar valor “TopicTweets”. Al campo “Client Name” hay que asignar “amoragaf”.

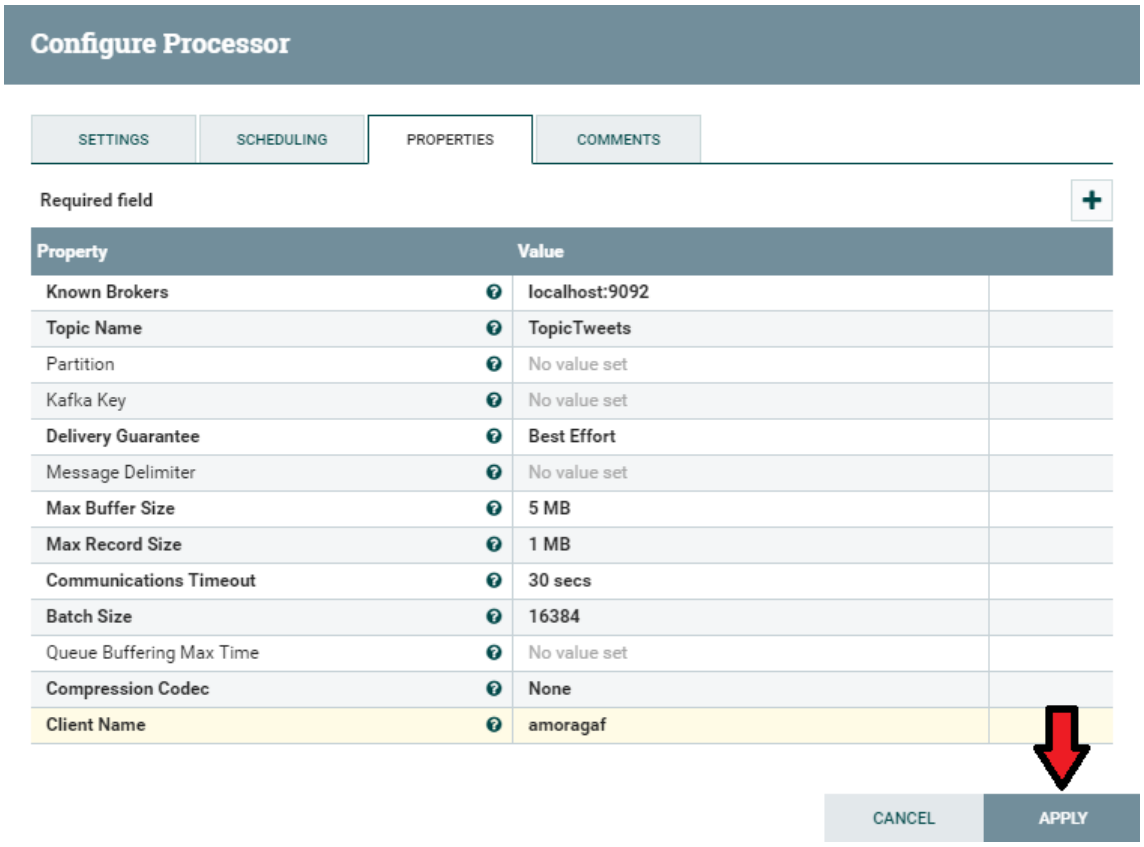


Ilustración 46 - Creación del flujo. De Twitter a Kafka

Los parámetros de configuración que han sido modificados han sido:

- **Known Brokers:** localhost:9092
- **Topic Name:** TopicTweets
- **Client Name:** amoragaf

Ahora que está configurado correctamente el processor “PutKafka”, hay que conectarlo con el processor de donde van a provenir los tweets ya filtrados. Para ello, como se ha hecho en anteriores ocasiones, se selecciona el processor “RouteOnAttribute” y el símbolo de conexión se arrastra hasta el processor destino, es decir, “PutKafka”.

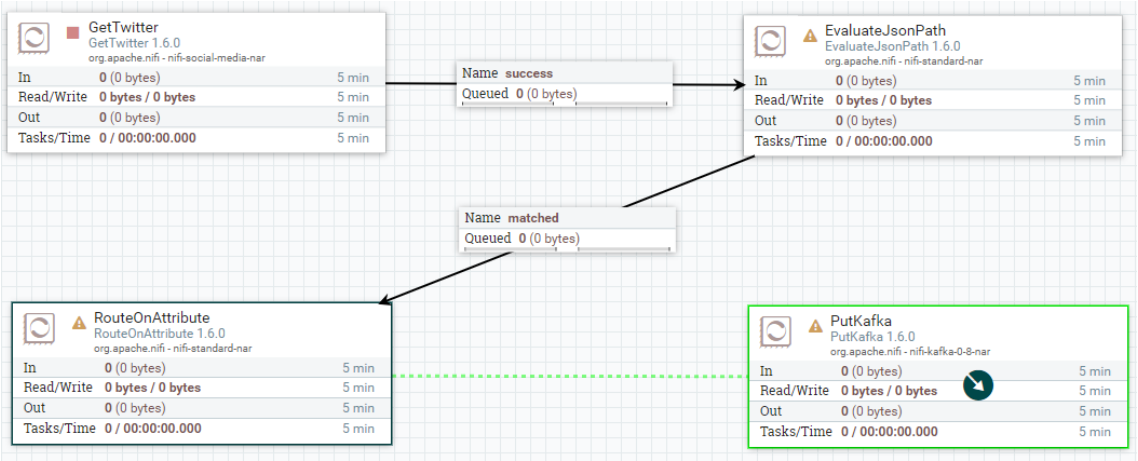


Ilustración 47 - Creación del flujo. De Twitter a Kafka

Aparecerá el menú de creación de la conexión. En el apartado “For relationships” se selecciona la opción “tweets”, es decir, la relación que va a llegar hasta putkafka son la de aquellos tweets filtrados. Posteriormente se hace click en el botón “Add”.

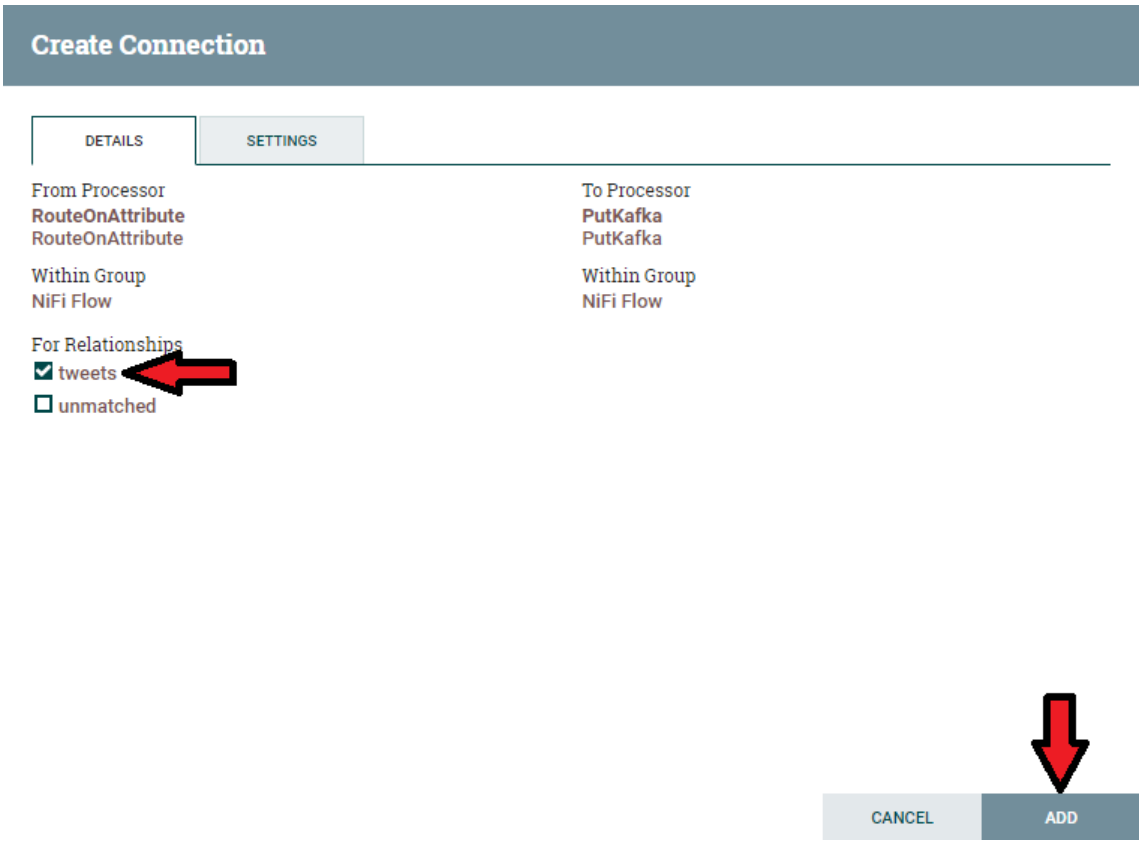


Ilustración 48 - Creación del flujo. De Twitter a Kafka

Una vez creada la última conexión que quedaba, ya se ha finalizado el flujo de trabajo necesario. Sin embargo, se observa que varios processors alertan que existen fallos de configuración en ellos.

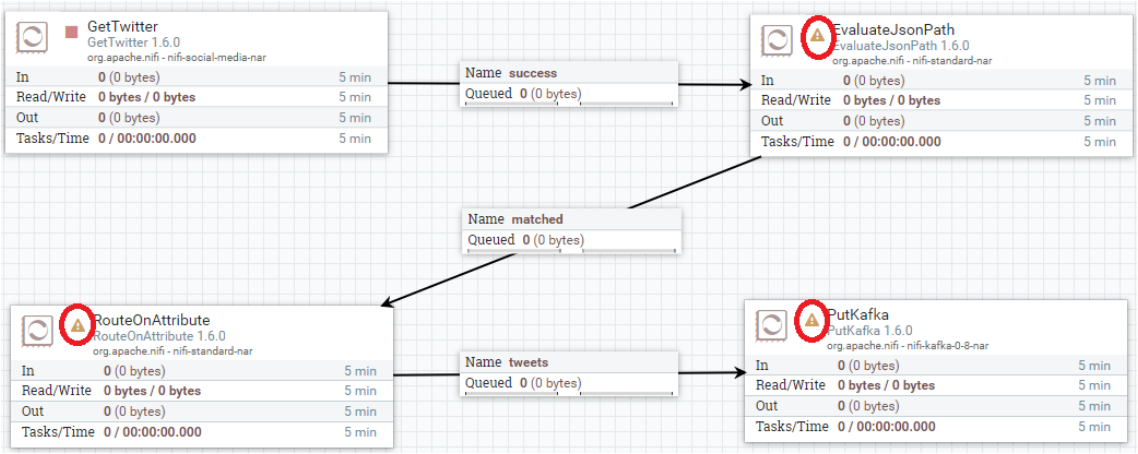


Ilustración 49 - Creación del flujo. De Twitter a Kafka

Esto se debe a que cuando se han realizado las conexiones entre los diferentes processor, esas conexiones se realizarán si se da un tipo de relación. Pero existen más tipos de escenarios, como puede ser por ejemplo, que el processor "EvaluateJsonPath" con la configuración que se ha introducido, de un fallo a la hora de analizar los Json. En ese caso, se daría una relación de fallo ("Failure"). En el caso de que se dé una de esas condiciones, nosotros lo que queremos es que los flowfiles no sigan recorriendo el flujo de trabajo. Para ello se indica que en el resto de tipos de relaciones finalice el flujo, así como, en el processor que representa el final del flujo de trabajo. Esto tenemos que hacerlo en los processors de "EvaluateJsonPath", "RouteOnAttribute" y "PutKafka" que son los que nos están dando fallo.

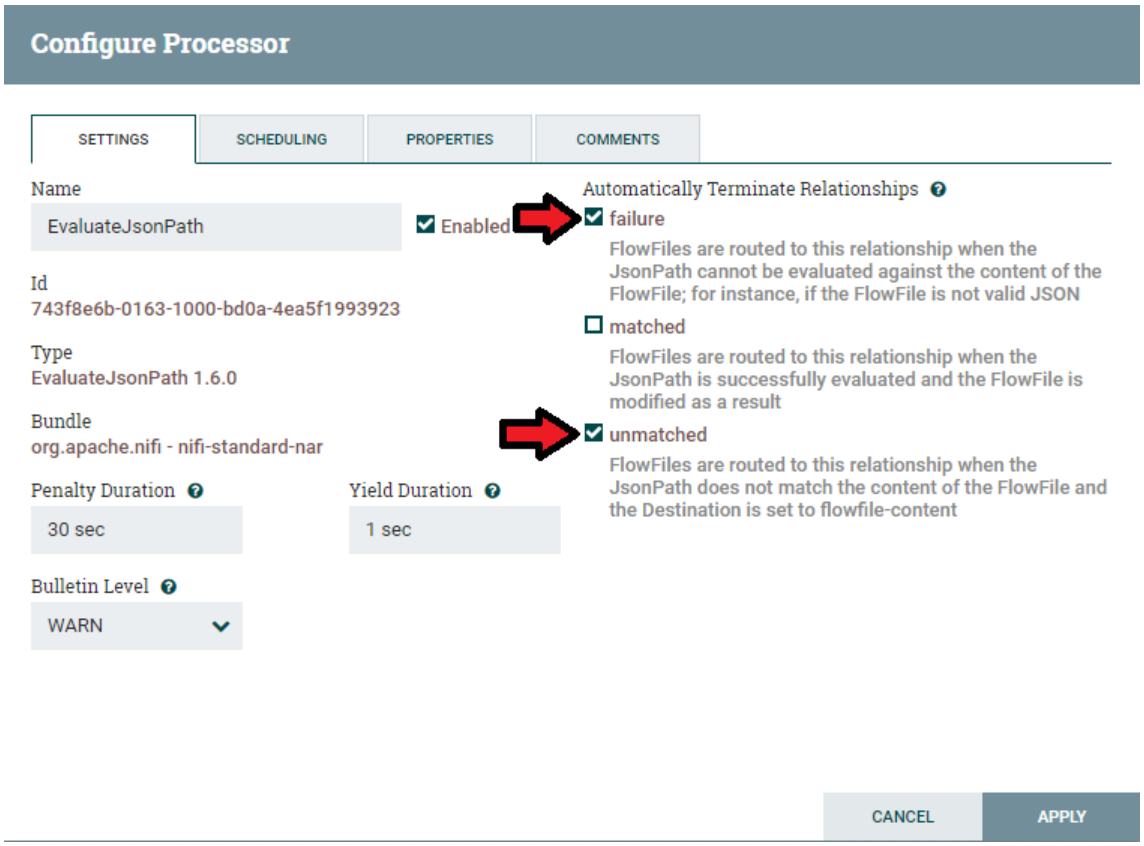


Ilustración 50 - Creación del flujo. De Twitter a Kafka

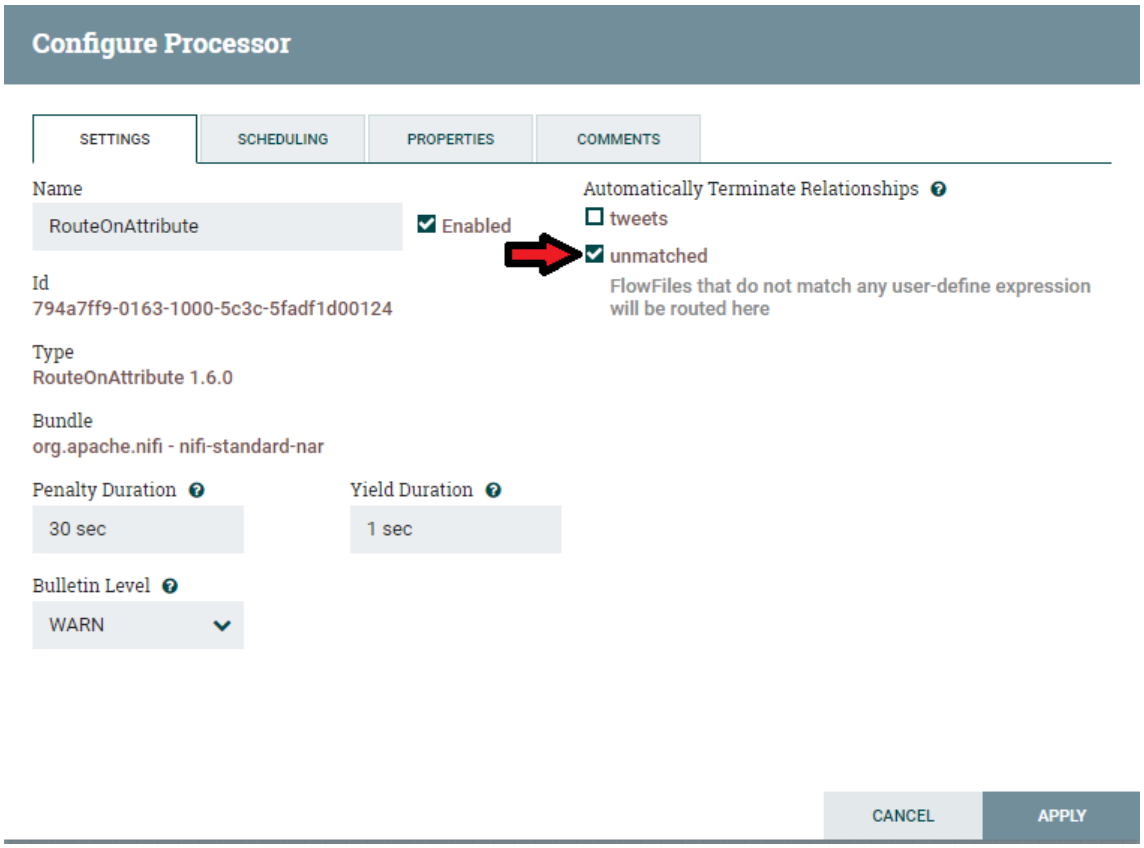


Ilustración 51 - Creación del flujo. De Twitter a Kafka

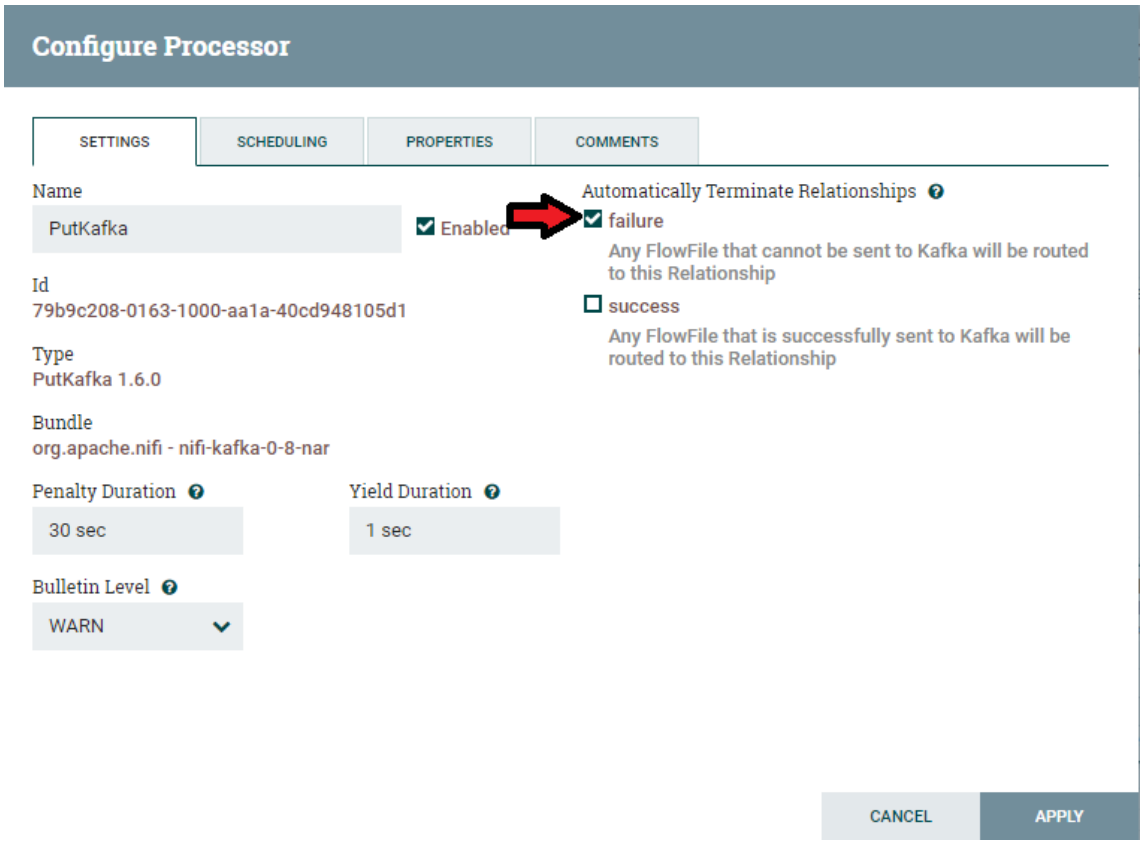


Ilustración 52 - Creación del flujo. De Twitter a Kafka

Ahora sí está el flujo de trabajo completo y perfectamente configurado. Se puede observar que todos los processors están configurados correctamente, esperando a ser iniciados.

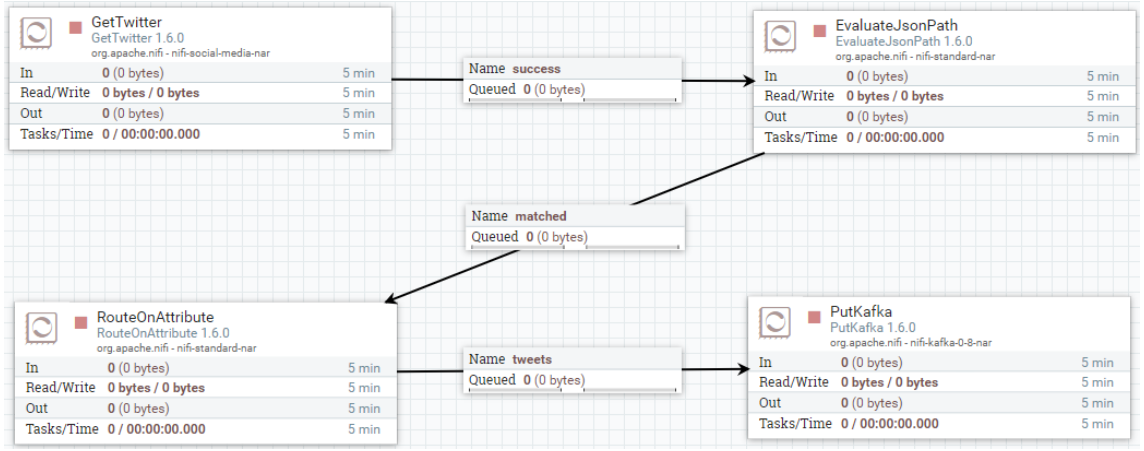


Ilustración 53 - Creación del flujo. De Twitter a Kafka

Para poner en funcionamiento el flujo de trabajo y conseguir el objetivo que hemos propuesto conseguir en esta herramienta, únicamente hay que iniciar los distintos processors que hay en nuestro flujo de trabajo.

Para ello, hacer click en cualquier parte de la cuadrícula de trabajo y seleccionar la opción “Start”. Con esto se inician todos los processors en vez de tener que ir arrancándolos uno a uno.

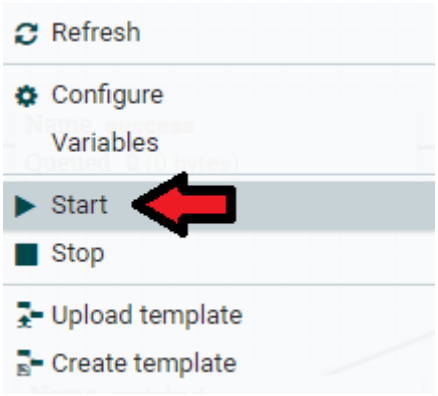


Ilustración 54 - Creación del flujo. De Twitter a Kafka

Una vez iniciados, hay que comprobar que los flowfiles recorren el flujo por completo y que el último de los processors que es el encargado de enviar los tweets a Kafka esté enviando realmente.

ALMACENAMIENTO CON APACHE KAFKA

Instalación de la herramienta

Se va a explicar los pasos necesarios para la instalación de Apache Kafka en nuestro ordenador. El primer paso es descargar la herramienta. Se dispone de la versión Apache Kafka 1.0.0 con la versión de Scala 2.11. Para descargar esta herramienta nos dirigimos a la pestaña de descargas de la web de Apache Kafka:

https://kafka.apache.org/downloads

Una vez aquí, seleccionar el tipo de archivo y la versión deseada. Por consiguiente, seleccionar los binarios de la versión 1.0.0 para la versión de Scala 2.11.

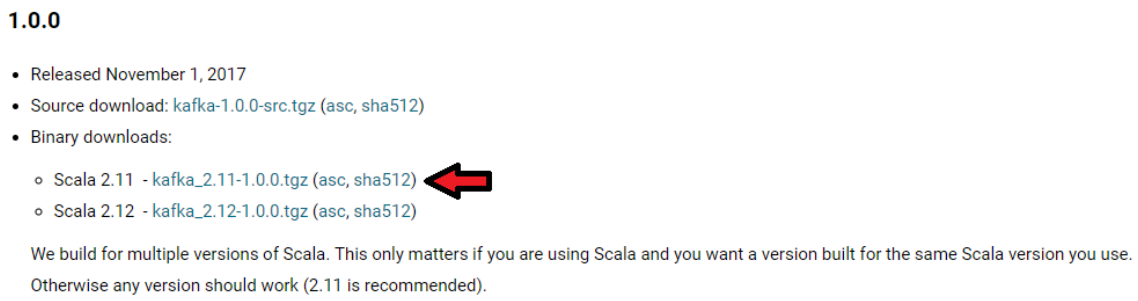


Ilustración 55 - Instalación de la herramienta

Se redirige a la web de Apache Software Foundation. Aquí se pincha en el enlace de descarga recomendado y acto seguido comienza la descarga de la herramienta.

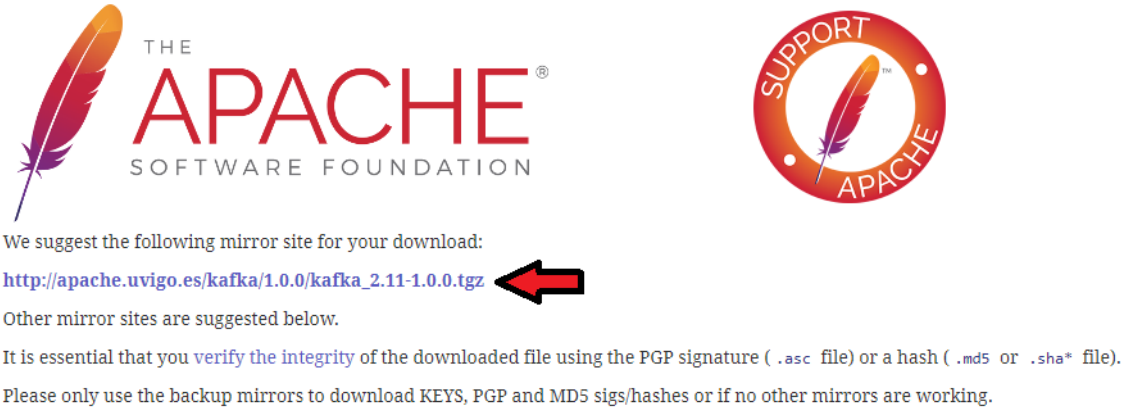


Ilustración 56 - Instalación de la herramienta

Una vez que ya está descargado el archivo comprimido, hay que descomprimirlo. En nuestro caso, para acceder en el futuro a la herramienta lo más rápido posible ya que cuando abres la consola es la ruta de trabajo que te aparece inicialmente, se ha decidido descomprimirlo en el directorio *C:\Users\amoragaf*

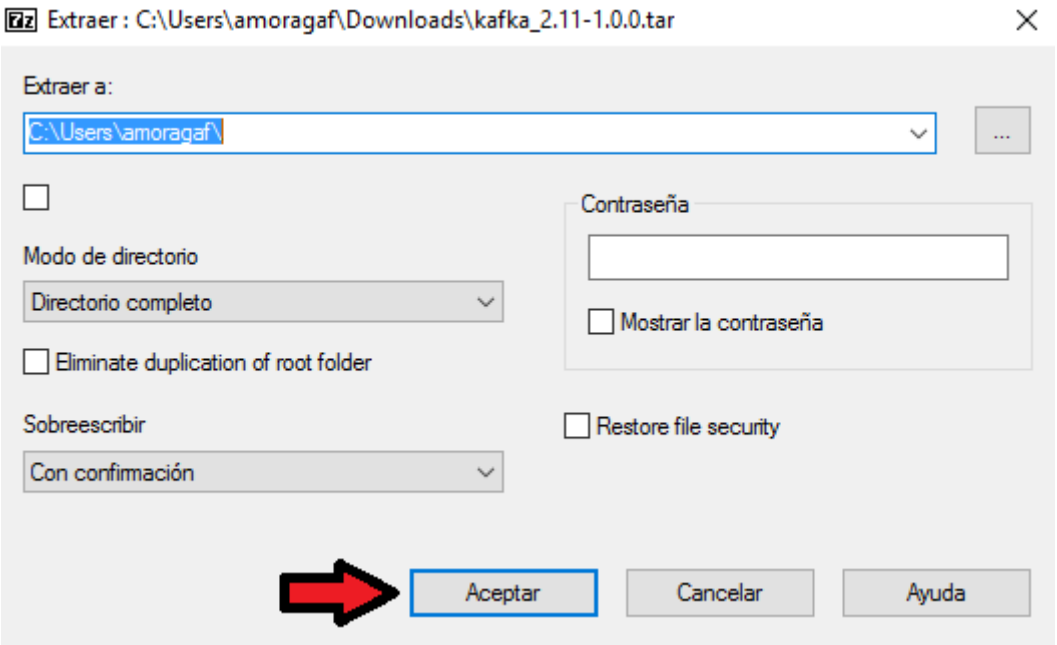


Ilustración 57 - Instalación de la herramienta

Una vez hecho esto, ya estarían todos los archivos necesarios para posteriormente levantar el servicio de Apache Kafka.

Configuración de un clúster de 3 nodos

La configuración que trae Apache Kafka por defecto es para trabajar con un único nodo. Si se levanta el servicio de Kafka con dicha configuración trabajaríamos con un único nodo y funcionaría correctamente pero sin obtener todas las ventajas de un servicio distribuido que puede llegar a ofrecer Kafka. Para descubrir las posibilidades que puede llegar Kafka y profundizar más en este trabajo que se ha realizado, se ha optado por configurar un clúster multi-nodo. Para ello se va a simular, ya que realmente se va a seguir trabajando con una única máquina en la que se van a levantar tres nodos de Kafka.

Para realizar este clúster multinodo, se va a crear la configuración de los nodos. Apache Kafka ya trae configurado un nodo por lo que hay que configurar los otros dos restantes. Para ello hay que dirigirse a la carpeta de configuración de Kafka. Que en nuestro caso se encuentra en:

C:\Users\amoragaf\kafka_2.11-1.0.0\config

Se localiza el archivo de configuración del nodo de Kafka. Este archivo es el llamado “server.properties”. Para crear la configuración de los otros dos nodos hay que copiar este archivo dos veces para su posterior modificación: “server-2.properties” y “server-3.properties”.

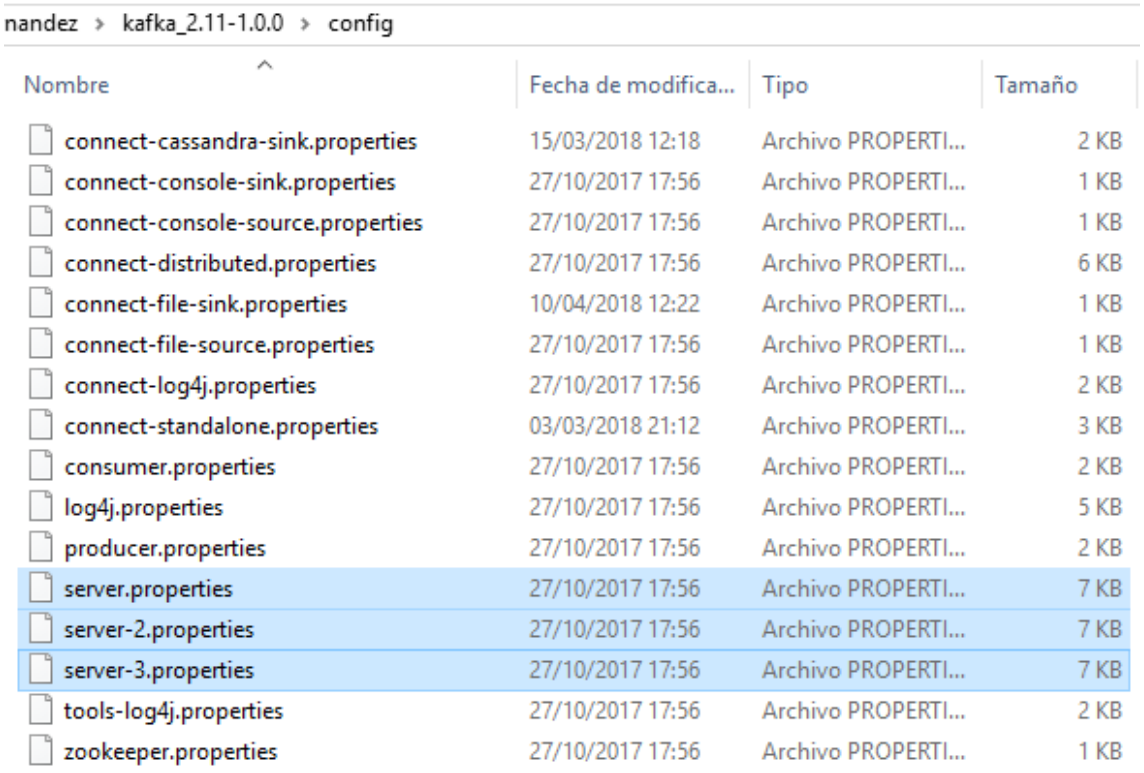


Ilustración 58 - Configuración de un clúster de 3 nodos

Para la modificación de los diferentes parámetros de configuración se va a necesitar un editor de texto, en nuestro caso se ha utilizado Notepad++. La configuración que se va a modificar respecto al nodo por defecto, va a hacer referencia al directorio donde se van a almacenar los archivos de registro, al identificador del nodo y el puerto del nodo.

Se comienza modificando la configuración del “server-2.properties”.

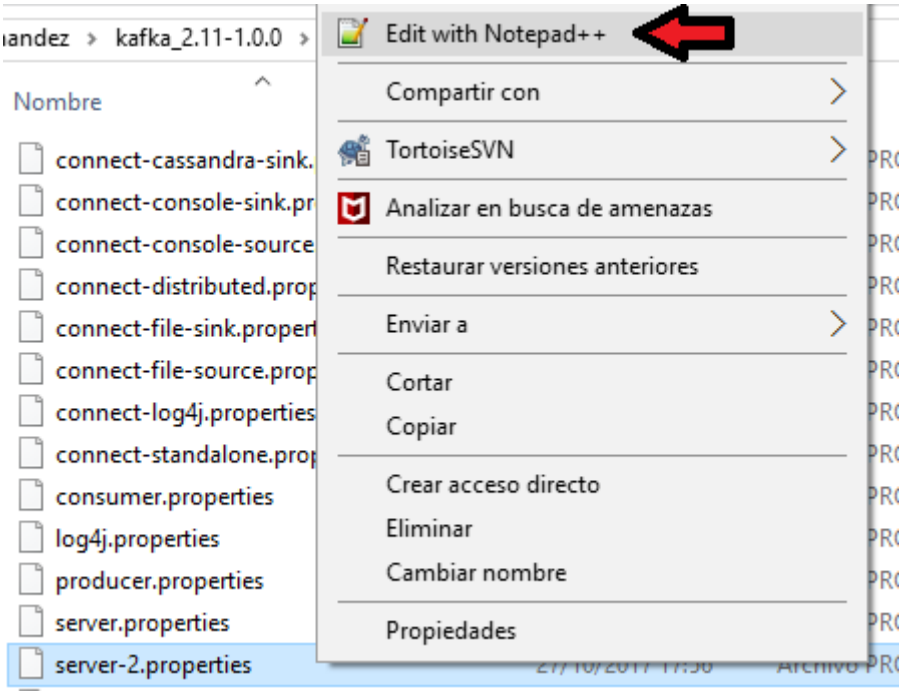


Ilustración 59 - Configuración de un clúster de 3 nodos

Se modifica el identificador del nodo. El nodo por defecto tiene el id=0 por lo tanto a este nuevo nodo se le da el valor de id=1.

```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1
```

Ilustración 60 - Configuración de un clúster de 3 nodos

Se indica el directorio donde almacenará los distintos archivos.

```
############################# Log Basics #############################

# A comma seperated list of directories under which to store log files
log.dirs=/tmp/kafka-logs-2
```

Ilustración 61 - Configuración de un clúster de 3 nodos

Por último, se indica cuál va a ser el puerto del nodo. En la configuración del nodo por defecto que trae Apache Kafka, la dirección del nodo viene comentada y coge la que es por defecto, que es con el puerto 9092.

Por lo tanto, se descomenta la línea en la que indica la dirección del puerto y se indica que el puerto es 9093.

```
############################# Socket Server Settings #############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://:9093
```

Ilustración 62 - Configuración de un clúster de 3 nodos

Ya se tendría la configuración de un nuevo nodo que se podría levantar. Para configurar el que sería nuestro tercer nodo, se repite el proceso. Para ello, se va a editar el archivo "server-3.properties".

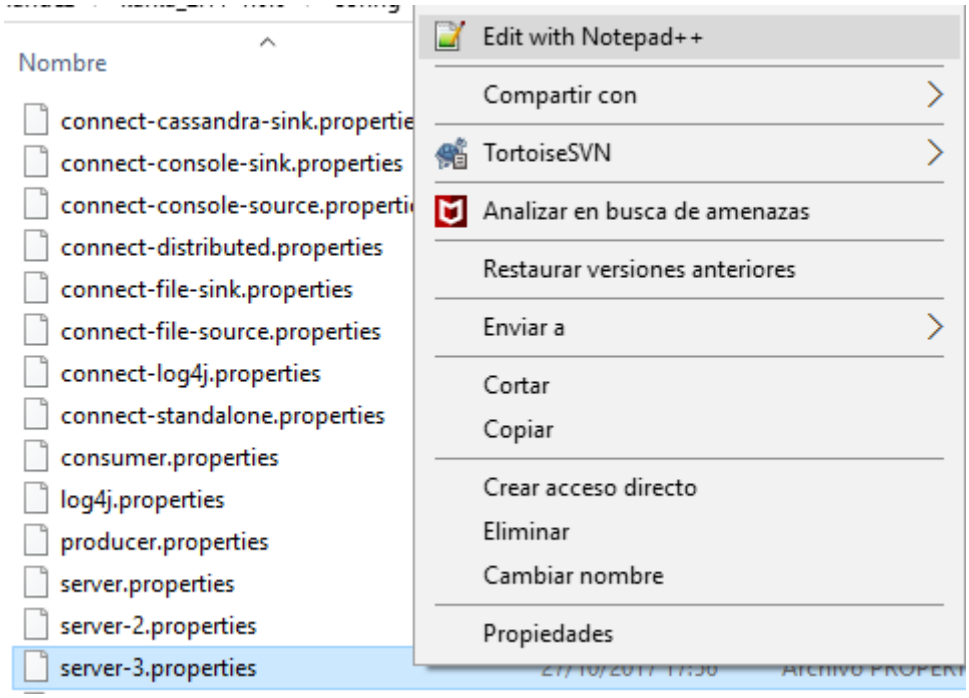


Ilustración 63 - Configuración de un clúster de 3 nodos

Se modifica el identificador del nodo. A este nodo se asigna el id=2.

```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=2
```

Ilustración 64 - Configuración de un clúster de 3 nodos

Se indica el directorio donde almacenará los distintos archivos.

```
############################# Log Basics #############################

# A comma seperated list of directories under which to store log files
log.dirs=/tmp/kafka-logs-3
```

Ilustración 65 - Configuración de un clúster de 3 nodos

Y se modifica el puerto del nodo

```
############################# Socket Server Settings #############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://:9094
```

Ilustración 66 - Configuración de un clúster de 3 nodos

Ya se ha creado la configuración de los distintos nodos para arrancar nuestra simulación de clúster multinodo.

Implementación del clúster

Teniendo todos los archivos de configuración correctamente, se está en disposición de iniciar la herramienta y poder ir almacenando todos los tweets que van a ir llegando desde la salida de Nifi gracias al processor “PutKafka”. Puntualizar que el trabajo que se va a realizar con Apache Kafka va a ser a través de línea de comandos.

Para levantar los tres nodos del clúster, primero se debe iniciar Zookeeper. Zookeeper es el encargado de la gestión, administración de los distintos nodos de Kafka. El primero de los pasos cuando se quiere levantar Kafka, ya sea un nodo, dos nodos o n nodos es iniciar Zookeeper ya que es el encargado de coordinar los nodos.

Para arrancar Zookeeper, se abre la consola de Windows. Para acceder se pulsan las teclas (⊞ Win) + R que abre el comando Ejecutar. Se escribe “CMD” y se pulsa aceptar.

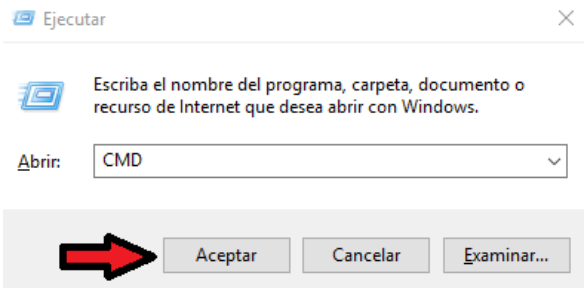


Ilustración 67 - Implementación del clúster

Una vez abierta la consola, hay que dirigirse al directorio donde se tiene Apache Kafka. Para ello se introduce en la consola:

cd kafka_2.11-1.0.0

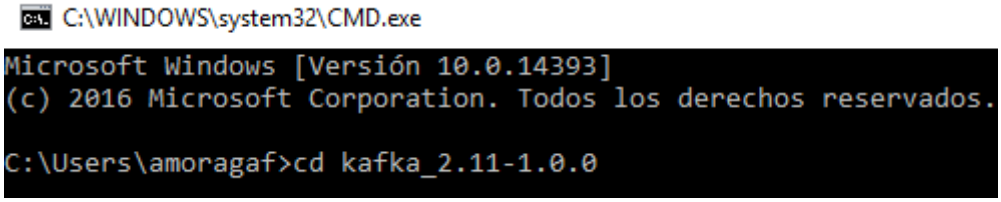


Ilustración 68 - Implementación del clúster

Una vez en el directorio de Kafka, a través de línea de comandos se ejecuta Zookeeper junto a su configuración correspondiente. Para ello se introduce en la terminal:

.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

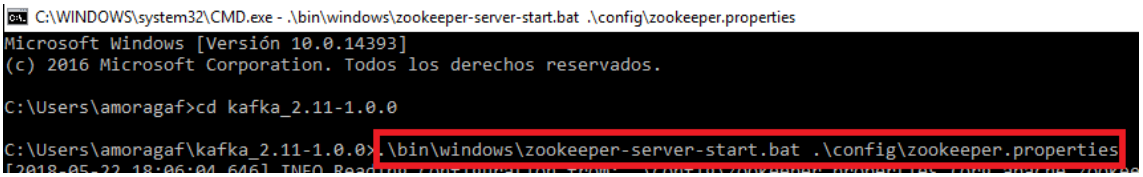


Ilustración 69 - Implementación del clúster

Cuando Zookeeper haya arrancado correctamente, se pueden levantar los diferentes nodos de Kafka, que serán coordinados por Zookeeper. Para arrancar los tres nodos, únicamente se debe seleccionar la configuración que corresponde a cada uno de los diferentes nodos.

NODO 1

Abrir una terminal y dirigirse al directorio donde está Apache Kafka. Una vez ahí se introduce la siguiente instrucción:

.\bin\windows\kafka-server-start.bat .\config\server.properties

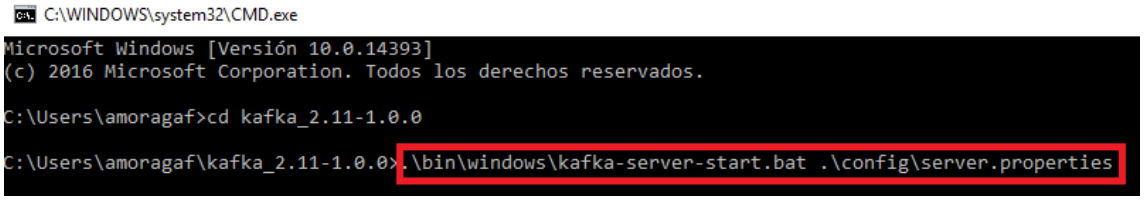


Ilustración 70 - Implementación del clúster

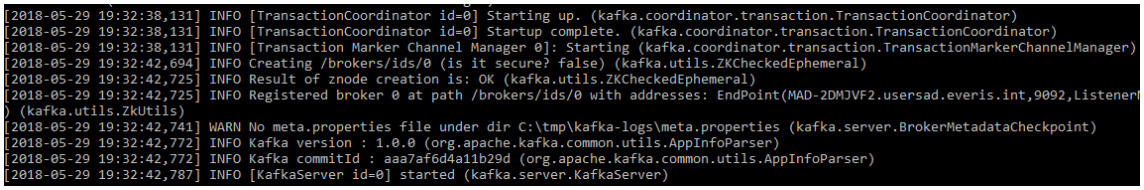


Ilustración 71 - Implementación del clúster

Nodo 2

Abrir una terminal y dirigirse al directorio donde está Apache Kafka. Una vez ahí se introduce la siguiente instrucción:

.\bin\windows\kafka-server-start.bat .\config\server-2.properties

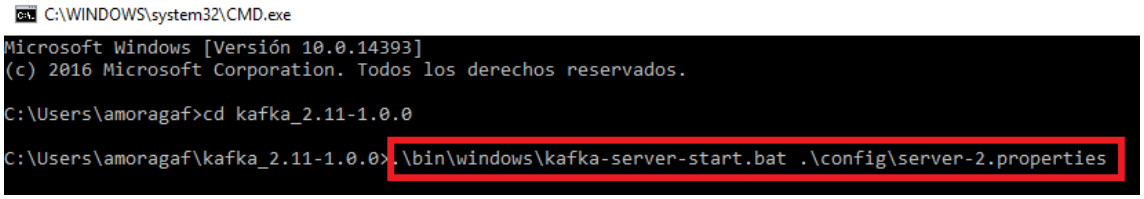


Ilustración 72 - Implementación del clúster

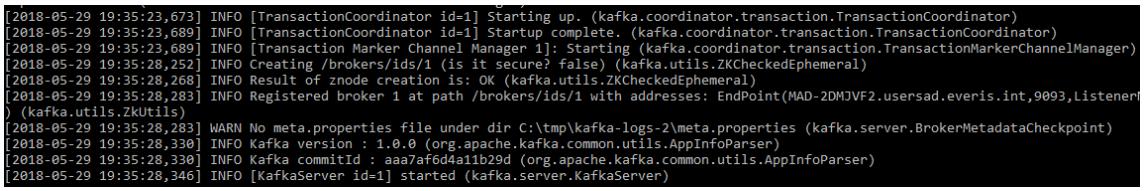


Ilustración 73 - Implementación del clúster

Nodo 3

Abrir una terminal y dirigirse al directorio donde está Apache Kafka. Una vez ahí introducir la siguiente instrucción:

.\bin\windows\kafka-server-start.bat .\config\server-3.properties

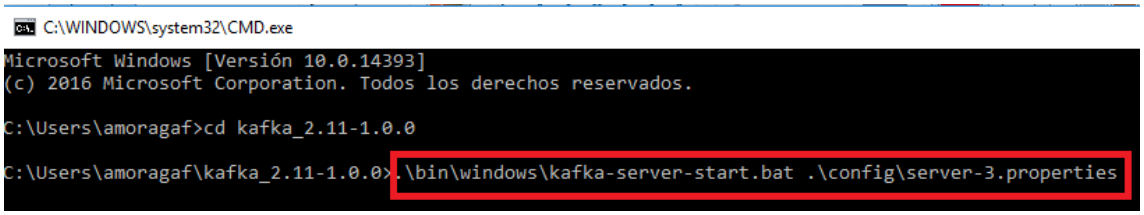


Ilustración 74 - Implementación del clúster

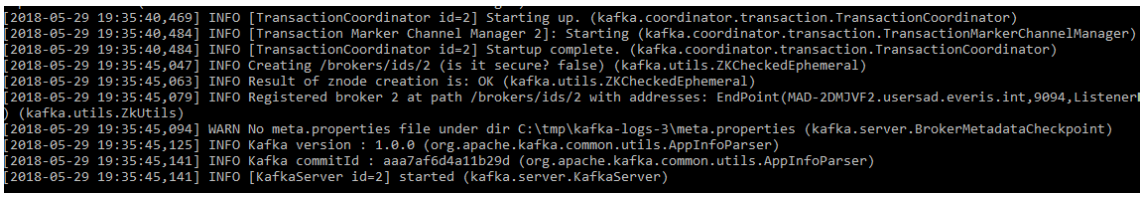


Ilustración 75 - Implementación del clúster

Creación del Topic y del Consumer

En estos momentos se tiene Zookeeper funcionando y gestionando los tres nodos de Kakfa. El siguiente paso es crear un topic. El topic funciona como una cola. En esta cola de Kafka se irán almacenando los tweets que vayan saliendo de Nifi.

Se va a crear un topic con factor de replicación = 3. Esto significa que el topic y su información, van a ser replicados en los tres nodos Kafka que se tienen levantados. Obviamente no se puede poner un factor de replicación mayor al número de nodos que hay disponibles.

Se debe asegurar que el nombre del topic es el mismo que el que le indicamos en Nifi en el proccesor “PutKafka”. Para crear este topic se abre una nueva consola, se va al directorio de Apache Kafka y se pone la siguiente instrucción.

.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 3 --partitions 1 --topic TopicTweets

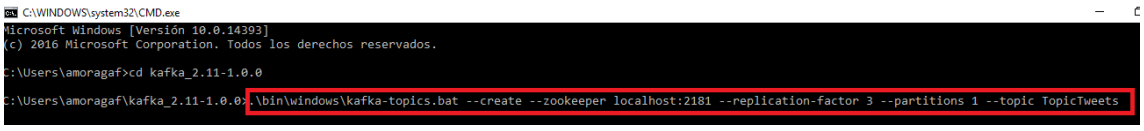


Ilustración 76 - Creación del Topic y del Consumer

Ahora que ya está creado el topic, si se inicia el flujo de Nifi, todos los registros se deberían ir almacenando en el topic de Kafka. Más adelante obviamente vamos a poder visualizar esos tweets y comprobar que el recorrido del dato entre las diferentes herramientas está siendo el correcto. Sin embargo se tiene la opción de hacer esa comprobación visual desde la misma herramienta de Kafka. Kafka ofrece la opción de crear un consumer que escuchará al topic y mostrará a tiempo real lo que va llegando al topic.

Para crear este consumer, se abre una nueva terminal, se va al directorio de Apache Kafka y se introduce la siguiente instrucción, indicándole el nombre del topic que debe consumir.

.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic TopicTweets --from-beginning

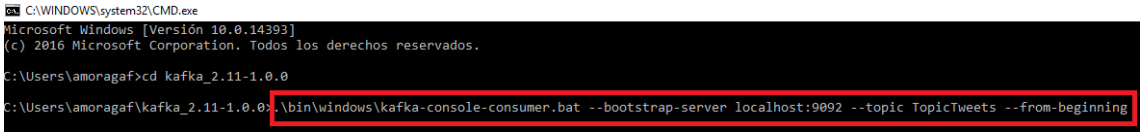


Ilustración 77 - Creación del Topic y del Consumer

Gracias a este consumidor de Kafka, se va a poder ir observando todos los datos que le estén llegando al topic que se estén consumiendo en ese momento. Se comprueba que la ingesta por parte de Nifi está siendo la correcta y se están almacenando adecuadamente en el topic deseado como se ve en la siguiente imagen.

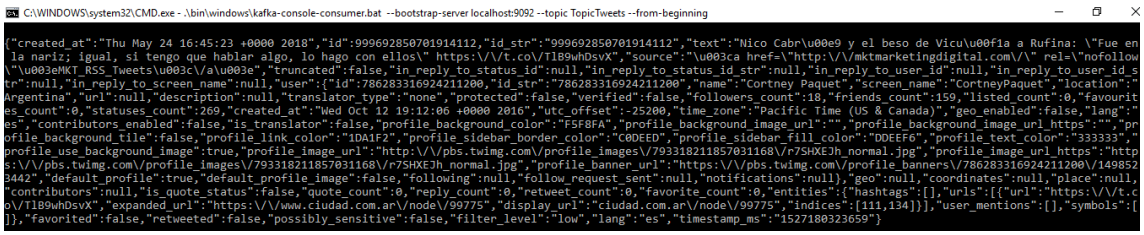


Ilustración 78 - Creación del Topic y del Consumer

Tolerancia a fallos del clúster

Se sabe que se dispone un clúster con diferentes nodos, pero de esos nodos no se tiene demasiada información. Para conocer qué es lo que está haciendo cada nodo en relación con el topic se ejecuta el comando “describe topics”.

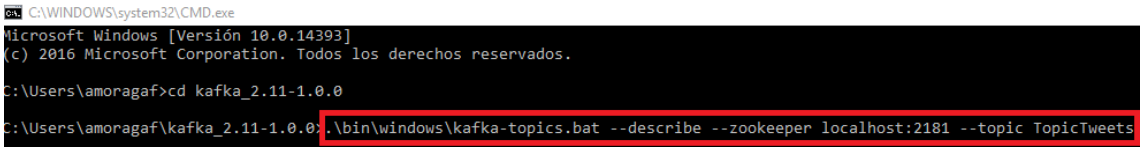


Ilustración 79 - Tolerancia a fallos del clúster

La salida que ofrece este comando en nuestro clúster es:

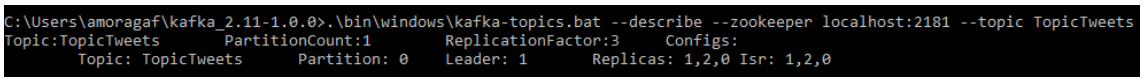


Ilustración 80 - Tolerancia a fallos del clúster

Se va a explicar la información que ofrece el output de este comando. La primera línea ofrece la información de las particiones del topic, como el topic se ha creado con una única partición, aparece solo una línea de información. Posteriormente ya aparece información del nodo en relación con cada uno de los nodos.

- “Leader”: Es el nodo responsable de todas las lecturas y escrituras de cada partición. Cada nodo será el líder de una parte de las particiones seleccionadas al azar.
- “Replicas”: Es la lita de nodos que replican el registro de esta partición, independientemente de si son el “leader” o incluso si están actualmente levantados.
- “Isr”: Conjunto de réplicas sincronizadas que están activas y dependen del líder. Es decir nodos sincronizados en los que la información está replicada. Como la información esta replicada y sincronizada en los distintos nodos, en el apartado anterior se podía hacer el consumidor de cualquiera de los nodos sincronizados que sería indiferente.

Ahora se va a demostrar la tolerancia al fallo de nuestro clúster. De los nodos de nuestro clúster, el nodo con id=1 está actuando como líder, así que, se va a eliminar ese nodo. Para realizar esta acción primero se debe saber id del proceso que le asigna Windows al nodo.

> wmic process where "caption = 'java.exe' and commandline like 'nombre del nodo'" get processid

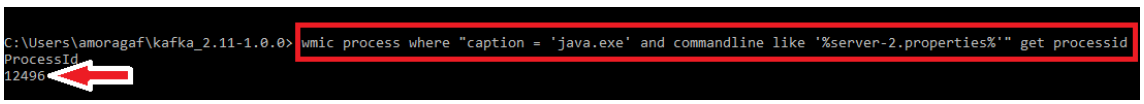


Ilustración 81 - Tolerancia a fallos del clúster

Una vez que se sabe el id, se puede “matar” ese nodo con la siguiente instrucción.

> taskkill /pid [nº de id que te devuelva] /f

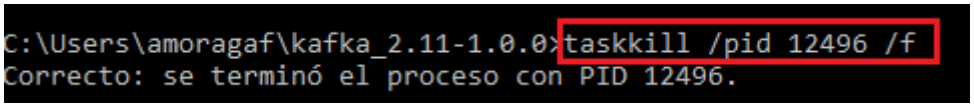


Ilustración 82 - Tolerancia a fallos del clúster

Se puede observar que ahora el nodo “leader” ha cambiado y ahora es uno de los que antes era su “esclavo”. Además a consecuencia de haber matado al nodo que actuaba de líder, ya no está en el conjunto de nodos replicados sincronizados (“Isr”)

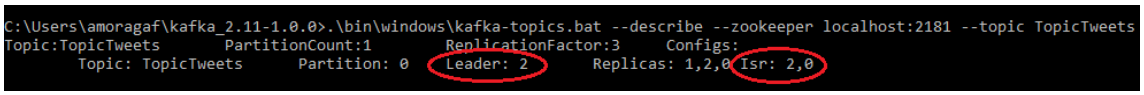


Ilustración 83 - Tolerancia a fallos del clúster

Pero como se trata de un sistema tolerante a fallos, si se arranca un consumer del topic en uno de los nodos que todavía están disponibles, a pesar de que el nodo leader que se encargó de las escrituras inicialmente está inactivo, se observa que se puede seguir consumiendo los datos perfectamente.

Se comprueba con el nodo de id=0

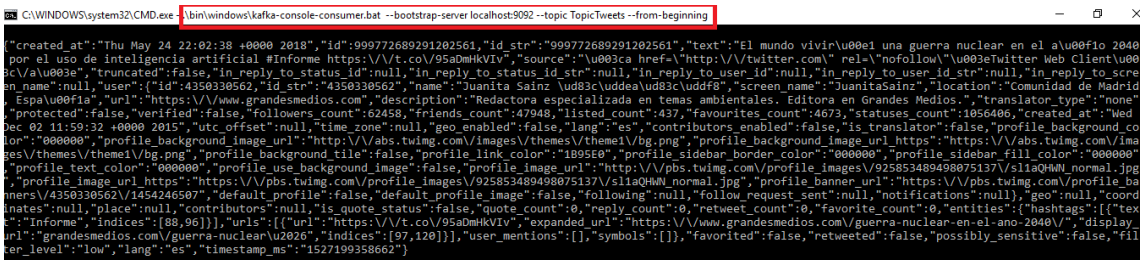


Ilustración 84 - Tolerancia a fallos del clúster

STREAMING CON KAFKA STREAMS

Una vez que se tienen almacenados todos los datos que se desean en Kafka, se necesita que esos datos estén disponibles para su posterior tratamiento y procesamiento en Spark. Aquí comienza una labor de investigación para saber cuál es el la mejor opción para nuestro caso en particular. Viendo diferentes escenarios de otros desarrollos Big Data, cuando es necesario utilizar los datos almacenados en un topic de Kafka para llevarlo a otro programa, base de datos, entorno o cualquier otra herramienta que vaya a necesitar los datos de Kafka, una de las opciones es utilizar Kafka Connect.

Kafka Connect es una herramienta para transferir datos de manera fiable y escalable entre Apache Kafka y otros sistemas. Ofrece la posibilidad de definir conectores de forma rápida y sencilla que son capaces de mover grandes cantidades de datos hacia Kafka o de Kafka hacia afuera.

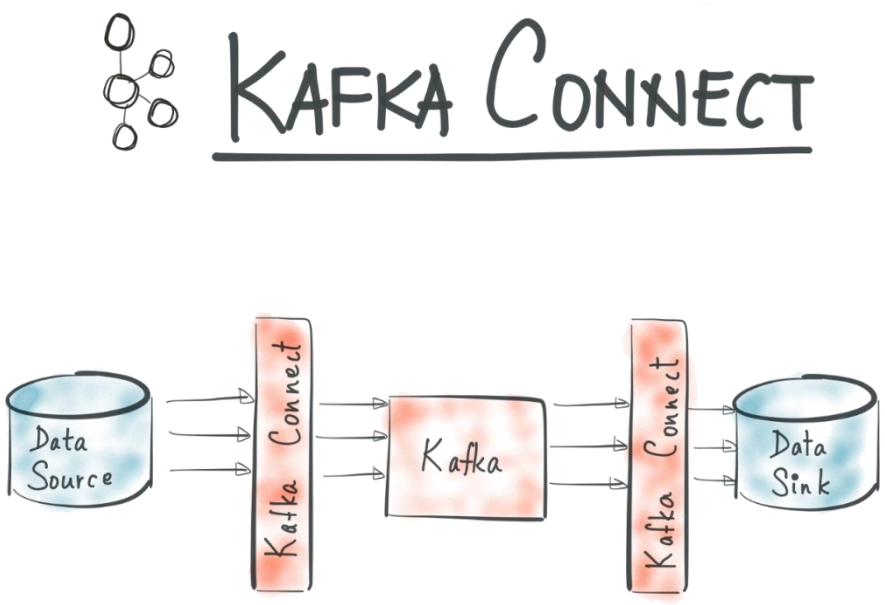


Ilustración 85 - Funcionamiento de kafka Connect

Kafka Connect puede ingestar bases de datos completas o recopilar métricas de todos sus servidores de aplicaciones en topics de Kafka, haciendo que los datos estén disponibles para el procesamiento de flujo con baja latencia. Además de la ingesta desde otros sistema, Apache Connect ofrece la funcionalidadd de realizar la exportación de los datos que están en los topic de Kafka en sistemas secundarios de almacenaiento y consulta o en sistemas por lotes para el análisis offline.

Sin embargo, una vez que se empieza a trabajar con Kafka Connect en nuestro caso real, nos dimos cuenta que no era la herramienta ideal. Existe la implementación de Kafka Connect predeterminadas para multiples sistemas, como pueden ser Cassandra, Elasticsearch o MongoDB, pero no existe una implementación propia para Apache Zeppelin que es la herramienta a la que se quiere exportar la información para su

posterior explotación. Aún así, se implementa nuestro propio conector de Kafka para Zeppelin, pero la información exportada no tenía el formato deseado. Se probó también que se exportase a un fichero en nuestro local, pero tampoco era el archivo deseado para posteriormente hacerle un procesamiento Spark.

Dado que esta opción no era viable implementarla en nuestra arquitectura, se sigue investigando otras opciones que dieran como resultado poder utilizar los datos que se tienen almacenado en Kafka para utilizarlos en nuestra herramienta de procesamiento y explotación. Finalmente se encontró con la solución, utilizar Kafka Streams. Para ello se crea un programa en Scala con el entorno de prpgramación Eclipse en el que se implementará la librería Kafka Streams para así obtener un output en un archivo en local que se podrá utilizar para su posterior procesamiento en Spark.

¿Qué es Kafka Streams?

Kafka Streams es una biblioteca cliente para crear aplicaciones y microservicios, en los que los datos de entrada o salida se almacenan en clústeres de Kafka. Combina la simplicidad de la escritura y la implementación de aplicaciones estándar de Java y Scala, con los beneficios de la tecnología del clúster de Kakfka.

Kafka Streams simplifica el desarrollo de aplicaciones aprovechando las bibliotecas de los producers y consumers de Kafka, ofreciendo las capacidades nativas de Kakfa para ofrecer paralelismo en los datos, coordinación a la hora de distribuir los datos, tolerancia a los fallos y simplicidad operativa. Se muestra como sería una aplicación al uso que utiliza la librería de Kafka Streams.

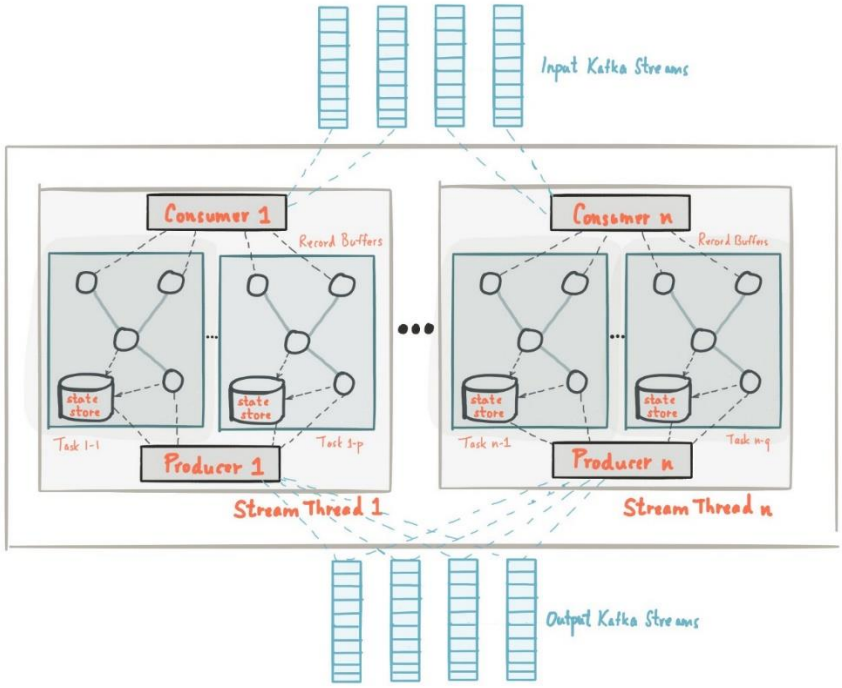


Ilustración 86 - Funcionamiento Kafka Streams

Las principales características de Kafka Streams son:

- Altamente escalable y tolerante al fallo.
- Posibilidad de implementarlo en Docker, máquinas virtuales, en la nube, etc.
- Igualmente viable para casos de uso pequeños, medianos y grandes.
- Totalmente integrado con la seguridad de Kafka
- Permite escribir aplicaciones Java estándar.
- No requiere de un clúster de procesado por separado.
- Se puede desarrollar en Mac, Windows y Linux.

Configuración del entorno de desarrollo

El desarrollo del código como ya se ha comentado anteriormente, se va a realizar en el entorno de desarrollo Eclipse. Sin embargo, como se va a programar en Scala y se trata de un proyecto Maven, hay que realizar una serie de configuraciones adicionales antes de comenzar a desarrollar el código. La versión de Eclipse más normalizada en este momento es la de Eclipse Oxygen. Esta versión de Eclipse no está preparada para implementar código en Scala, por lo que se ha trabajado con el Scala IDE para Eclipse. Se puede obtener en el siguiente enlace:

http://scala-ide.org/download/sdk.html

Esta distribución contiene:

- Eclipse 4.7.1 (Oxygen)
- Scala IDE 4.7.0
- Scala 2.12.3 with Scala 2.11.11 and Scala 2.10.6
- Zinc 1.0.0
- Scala Worksheet 0.7.0
- ScalaTest 2.10.0.v-4-2_12
- Scala Refactoring 0.13.0
- Scala Search 0.6.0
- Scala IDE Play2 Plugin 0.10.0
- Scala IDE Lagom Plugin 1.0.0

Ahora ya se podría desarrollar nuestro código en lenguaje de programación Scala. Sin embargo, para un control de versiones de las diferentes librerías necesarias para implementar la herramienta Kafka Streams, se ha decido realizar un proyecto Maven. Gracias a Maven únicamente se tendrá que indicar las dependencias de las librerías que van a ser necesarias y Maven las gestiona automáticamente, evitando así, descargar los jar e introducirlos en nuestro proyecto pudiendo generar conflictos entre las diferentes librerías.

Para descargarnos la herramienta, se hace desde la página oficial del Apache Maven Project

https://maven.apache.org/download.cgi

Hay que dirigirse al directorio donde se ha guardado el archivo descargado. Una vez descomprimido, se guarda en la ubicación que se desee. Para nuestra comodidad se ha guardado en “C:\”.

Para poder utilizar Maven se necesita tener configuradas dos variables de entorno: MAVEN_HOME y PATH

Para configurar estas variables, se debe de ir a Este equipo -> (click derecho) -> Propiedades -> Configuración avanzada del sistema -> (Pestaña) Opciones avanzadas -> (Botón) Variables de entorno.

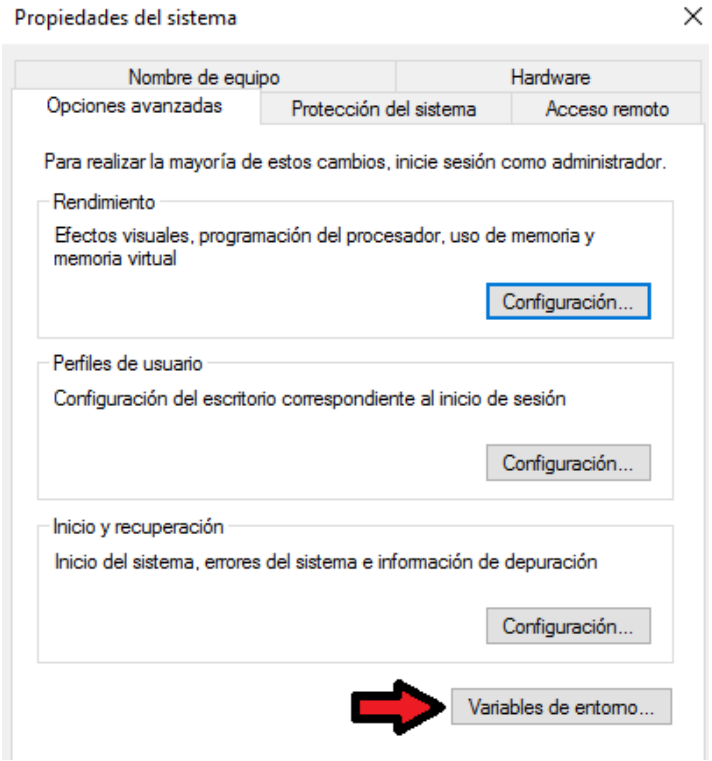


Ilustración 87 - Configuración del entorno de desarrollo

Las variables que se van a crear o editar serán variables del sistema, no variables del usuario.

Para M2_HOME se ingresa la ruta donde se descomprime maven.

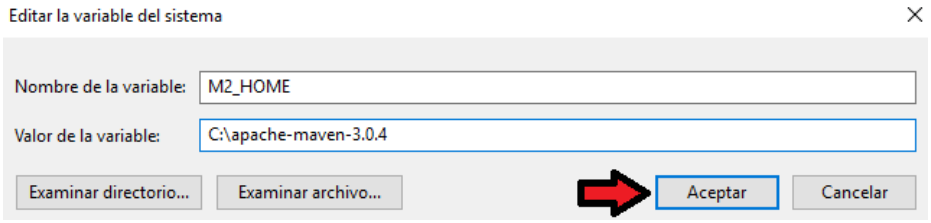


Ilustración 88 - Configuración del entorno de desarrollo

Ahora se tienen que editar nuestra variable PATH. Para ello se selecciona dicha variable y se pincha en editar. Una vez se está dentro, se hace click en Nuevo y se añade el siguiente valor:

%M2_HOME%\bin

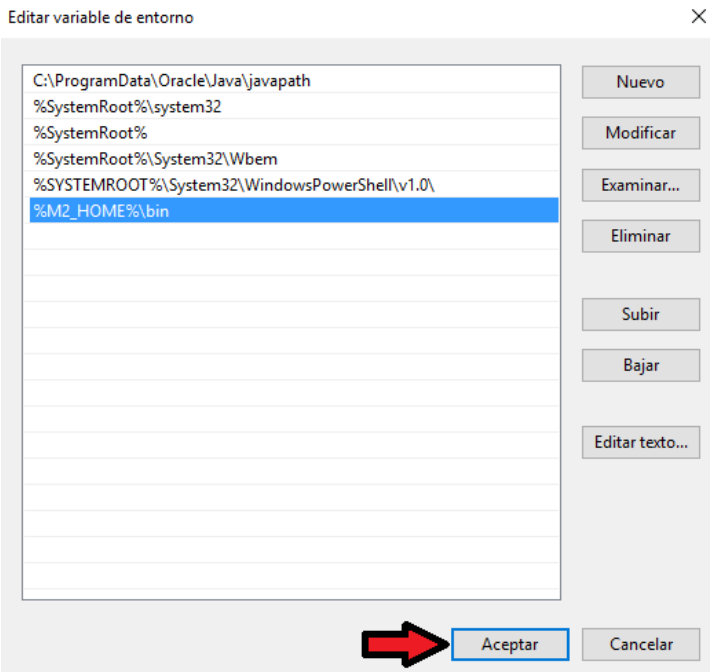


Ilustración 89 - Configuración del entorno de desarrollo

Después de esto, para comprobar que se ha realizado la instalación con éxito, se abre la consola y se ingresa el comando: mvn –versión

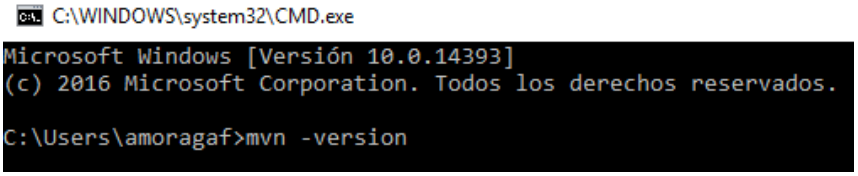


Ilustración 90 - Configuración del entorno de desarrollo

Debería de devolver lo siguiente:

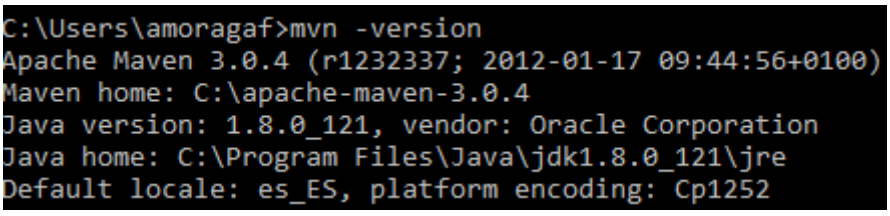


Ilustración 91 - Configuración del entorno de desarrollo

Configuración del Settings

Para un correcto funcionamiento de nuestro proyecto, se debe de cambiar la configuración del fichero settings que viene por defecto.

Para ello, dentro de nuestra carpeta “apache-maven-3.0.4” hay que dirigirse a la carpeta “conf”. Aquí se encuentra un único archivo llamado “settings.xml”.

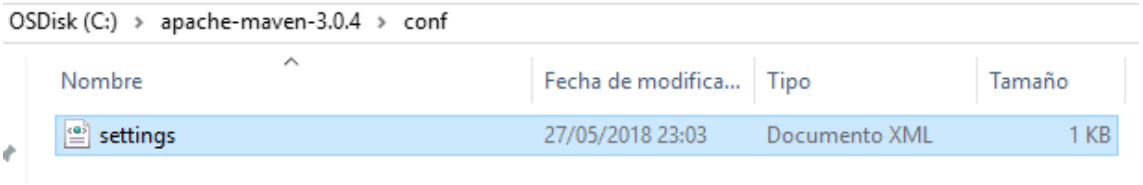


Ilustración 92 - Configuración del entorno de desarrollo

Se abre con nuestro editor de texto y se introduce la configuración deseada:

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                          https://maven.apache.org/xsd/settings-1.0.0.xsd">
      <localRepository/>
      <interactiveMode/>
      <usePluginRegistry/>
      <offline/>
      <pluginGroups/>
      <servers/>
      <mirrors/>
      <proxies/>
      <profiles/>
      <activeProfiles/>
</settings>
```

Ilustración 93 - Configuración del entorno de desarrollo

CONFIGURACIÓN ECLIPSE CON MAVEN

Una vez que se tiene configurado Maven correctamente en nuestro ordenador, se debe configurar Maven en Eclipse. Para ello dentro de nuestro Scala IDE for Eclipse, hay que dirgirse a Window -> Preferences -> Maven -> Installations. Hay que añadir nuestra instalación del Maven que se ha realizado anteriormente. Para ello se hace click en Añadir

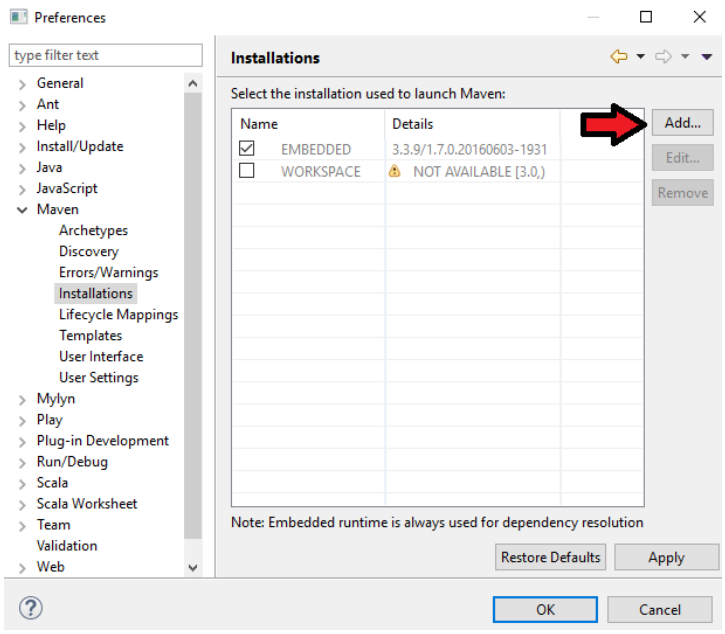


Ilustración 94 - Configuración del entorno de desarrollo

En este punto se debe indicar el directorio de nuestra instalación del Maven. Para ello se pulsa sobre Directory.

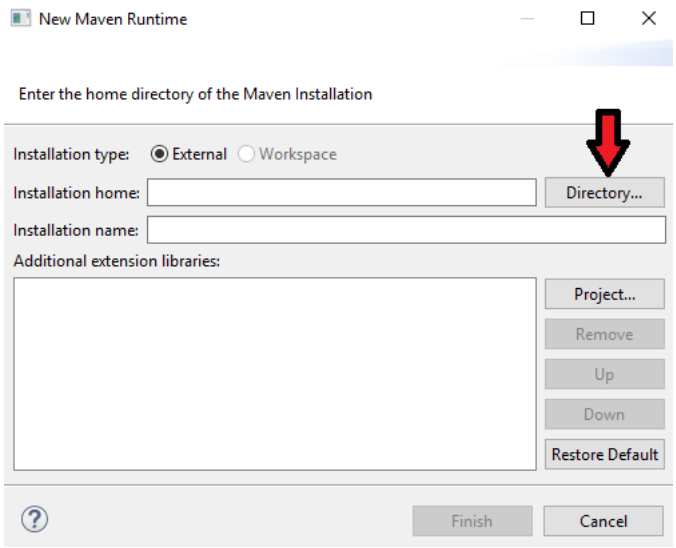


Ilustración 95 - Configuración del entorno de desarrollo

Se busca la carpeta donde está la instalación del Apache Maven anteriormente realizada, en nuestro caso *C:\apache-maven-3.0.4*

Una vez seleccionada la carpeta, se hace click en Finalizar.

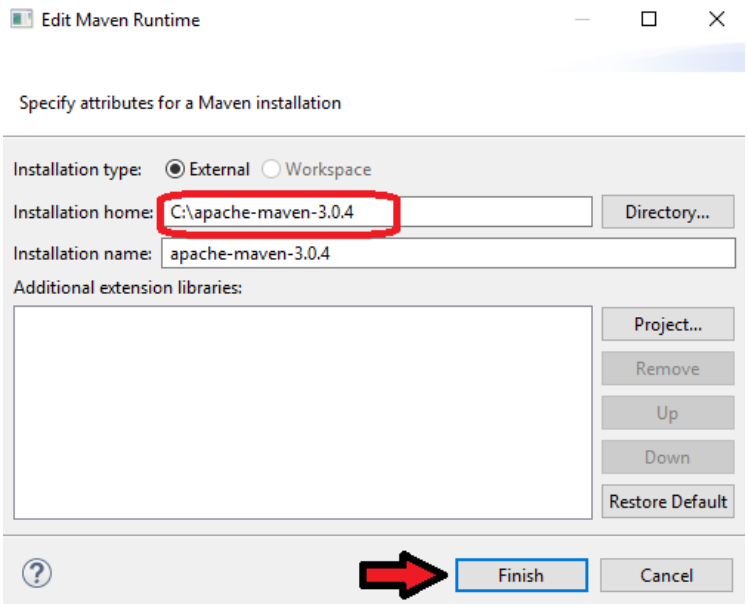


Ilustración 96 - Configuración del entorno de desarrollo

Hay que asegurarse que la instalación del Maven que se acaba de añadir es la que queda seleccionada. Una vez que está seleccionada, se hace click en Aplicar para que se haga efectivo el cambio.

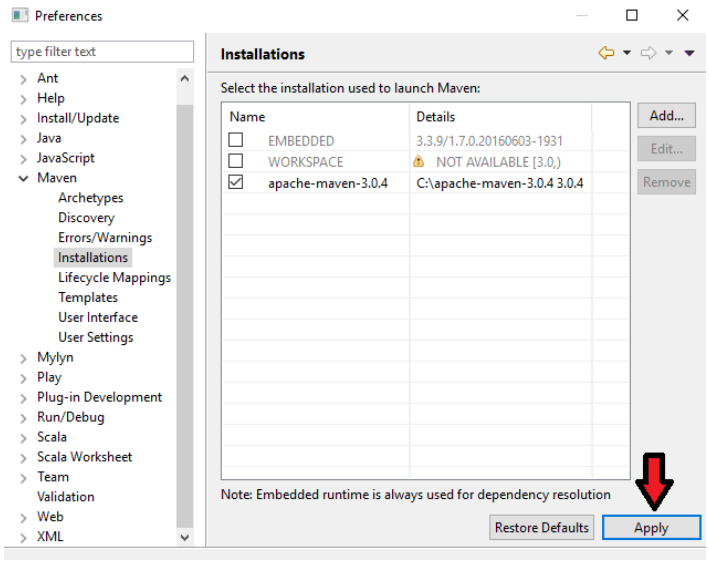


Ilustración 97 - Configuración del entorno de desarrollo

Ahora que está instalado, se debe de poner nuestra configuración del Maven. Para ello hay que dirigirse a Window -> Preferences -> Maven -> User Settings.

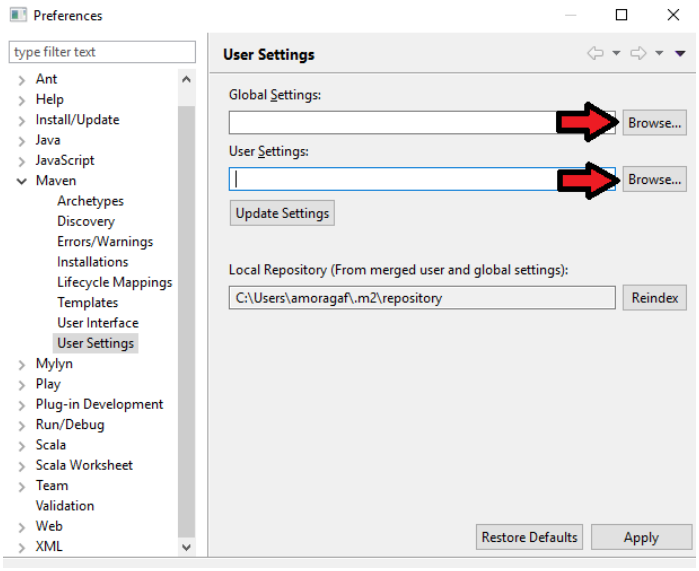


Ilustración 98 - Configuración del entorno de desarrollo

Tanto en Global Settings como en User settings se selecciona nuestro archivo settings.xml

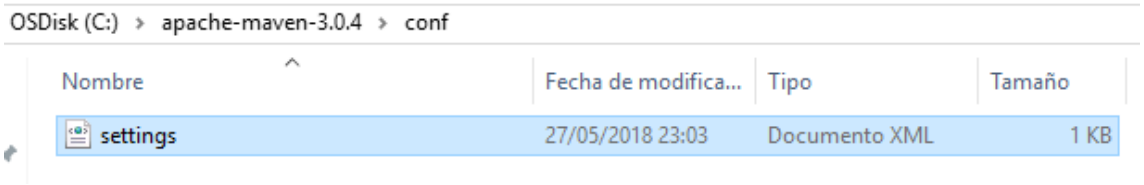


Ilustración 99 - Configuración del entorno de desarrollo

Una vez seleccionado nuestro archivo Settings en ambos campos, se hace click en Aplicar para que se hagan efectivos los cambios.

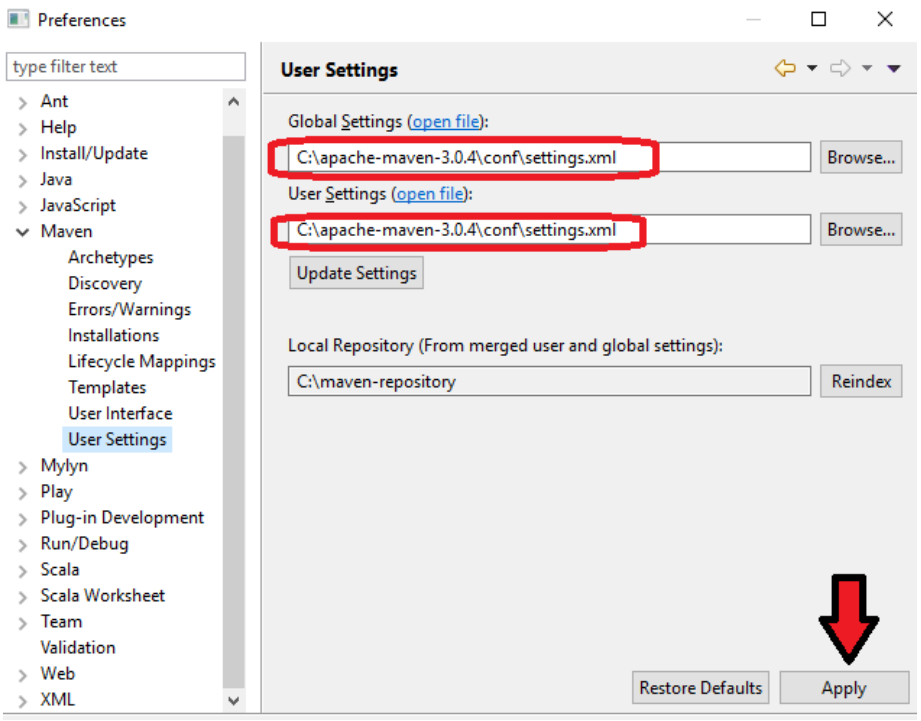


Ilustración 100 - Configuración del entorno de desarrollo

Maven por defecto tiene ya disponibles los arquetipos de los proyectos Java. Como se quiere crear un nuevo proyecto Scala Maven, hay que cargar los arquetipos que se van a necesitar. Para ello hay que dirigirse a Window -> Preferences -> Maven -> Archetypes. Una vez ahí se pincha en Add Remote Catalog

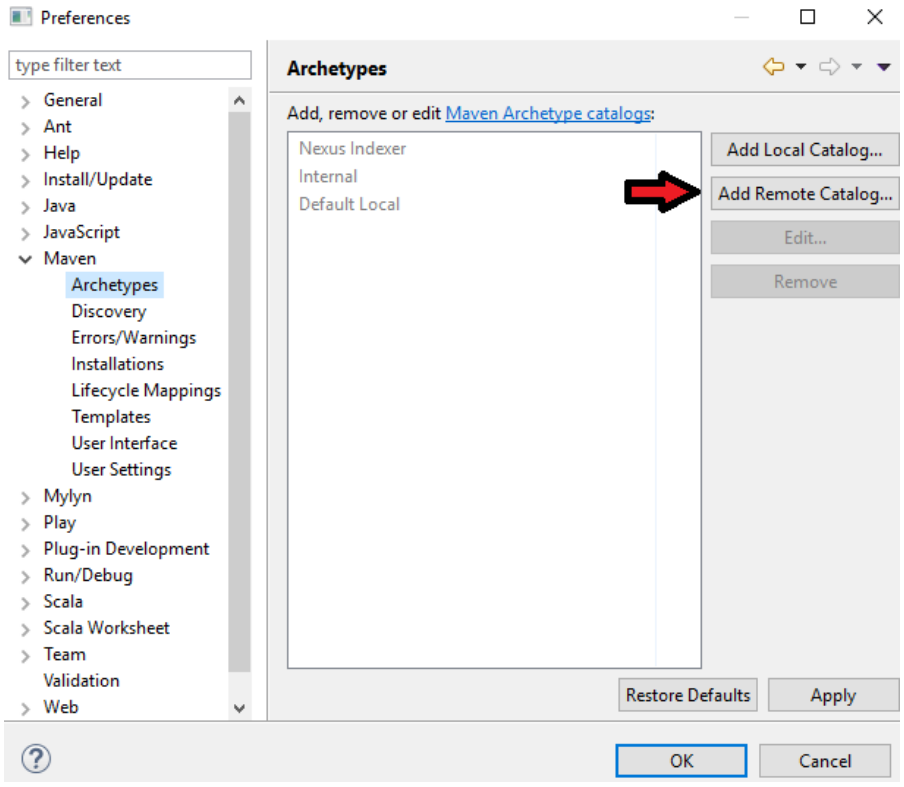


Ilustración 101 - Configuración del entorno de desarrollo

Se introduce la dirección del repositorio de Maven donde se encuentran los diferentes arquetipos, incluido el arquetipo de Scala que se va a utilizar para construir nuestro proyecto. El repositorio es:

http://repo1.maven.org/maven2/archetype-catalog.xml

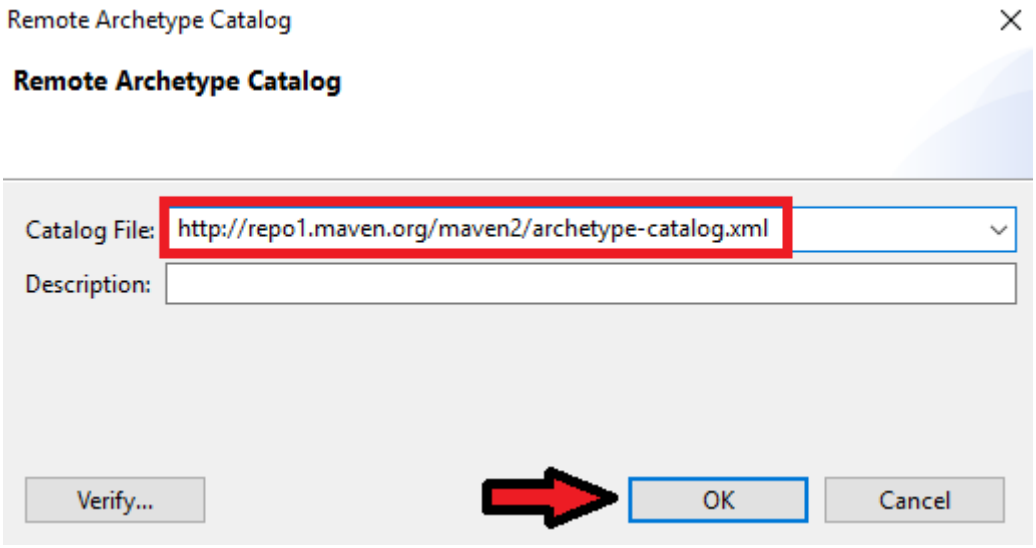


Ilustración 102 - Configuración del entorno de desarrollo

Ahora sí ya se tiene toda la configuración realizada en nuestro Scala IDE para empezar a desarrollar nuestro código para implementar la herramienta Kafka Streams.

Creación de proyecto Maven

Como ya se ha comentado antes, se va a construir un proyecto Maven en Scala. El motivo de esta elección es que es el lenguaje con mejor rendimiento cuando se está hablando de procesamiento Big Data es Scala. Una vez se comienza a programar esta implementación de Kafka Streams, surgió la necesidad de utilizar distintas librerías que tras descargarlas se podían añadir a nuestro proyecto Scala. A partir de ahí empezaron los problemas: unos jar venían con demasiadas librerías que eran innecesarias, faltaban las versiones que se necesitaban, se duplicaban librerías entre distntos jars, se creaban dependencias que entraban en conflicto, etc. Por ello, tras una investigación sobre posibles soluciones se llegó a la conclusión que la mejor solución era utilizar Maven. Maven es un gestor de dependencias que a través de un fichero Pom.xml construirá el proyecto con el arquetipo y dependencias deseadas.

Para la creación del proyecto Scala Maven, hay que dirigirse a File -> New -> Other. A continuación aparecerá una ventana emergente y después de seleccionar el desplegable de “Maven” se elige la opción “Maven Project” y se pulsa “Next”.

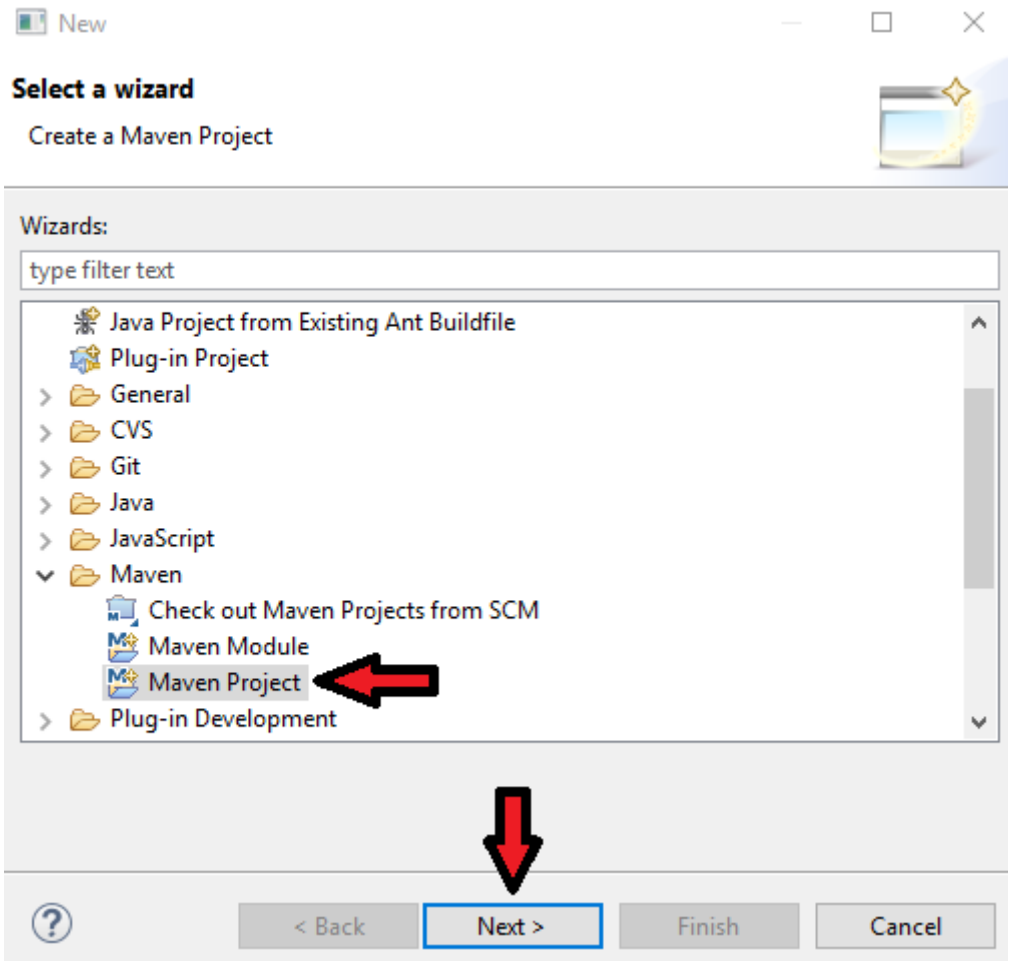


Ilustración 103 - Creación de proyecto Maven

Se comprueba que la opción de saltar el paso de la selección del arquetipo está desactivada y se indica la ruta en la que se quiere crear nuestro proyecto. Una vez hecho esto se hace click en “Next”.

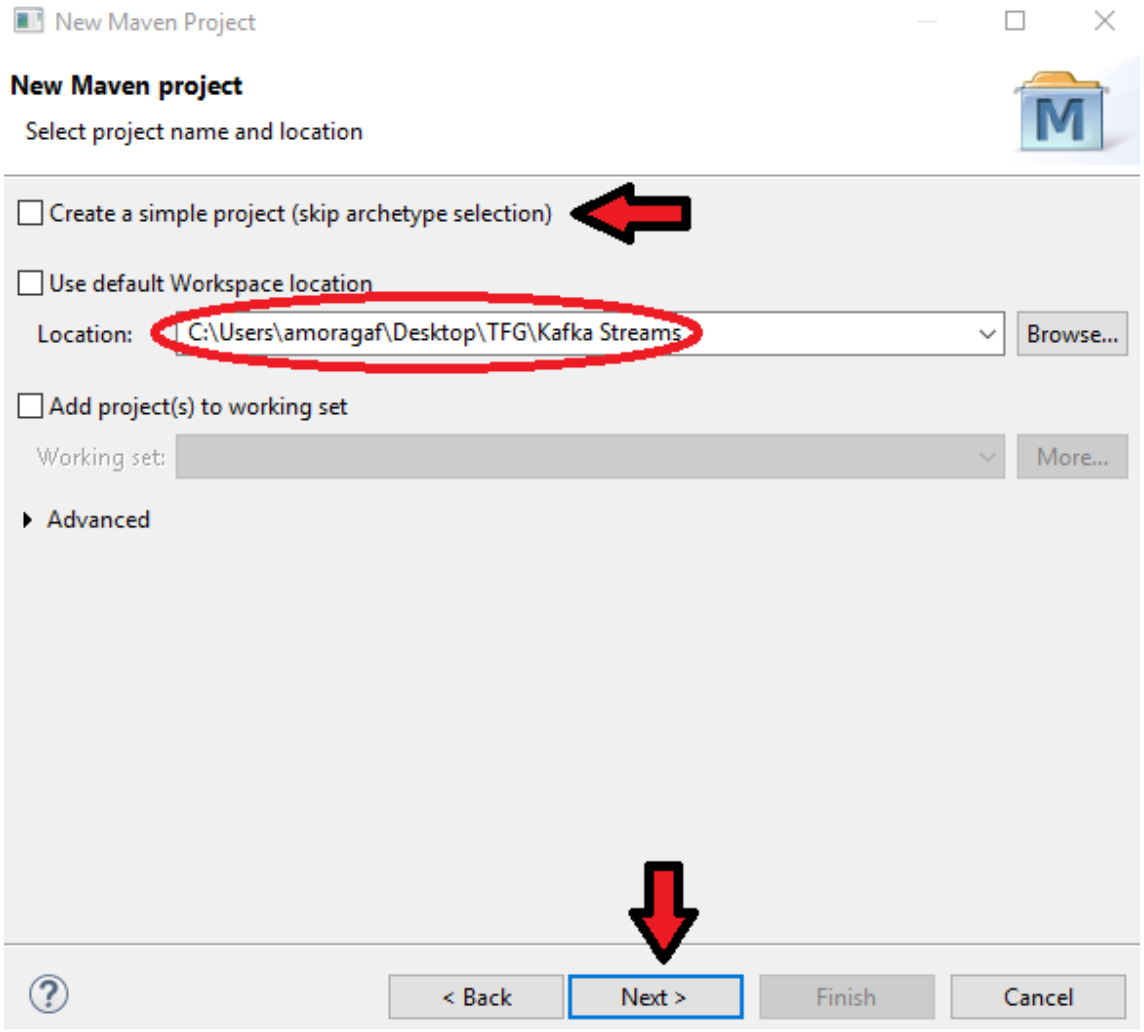


Ilustración 104 - Creación de proyecto Maven

En este paso se debe de seleccionar el arquetipo que se desea para nuestro proyecto. Se comprueba que el catálogo que se muestra está en la opción “All catalogs” para así mostrar los arquetipos que se han añadido de Scala anteriormente. Para mayor velocidad de búsqueda se va a filtrar los arquetipos por “scala-archetype”. Se selecciona el arquetipo con Goup Id “org.scala-tools.archetypes”, Artifact Id “scala-archetype-simple” y versión 1.3. Una vez seleccionado se pulsa en “Next”.

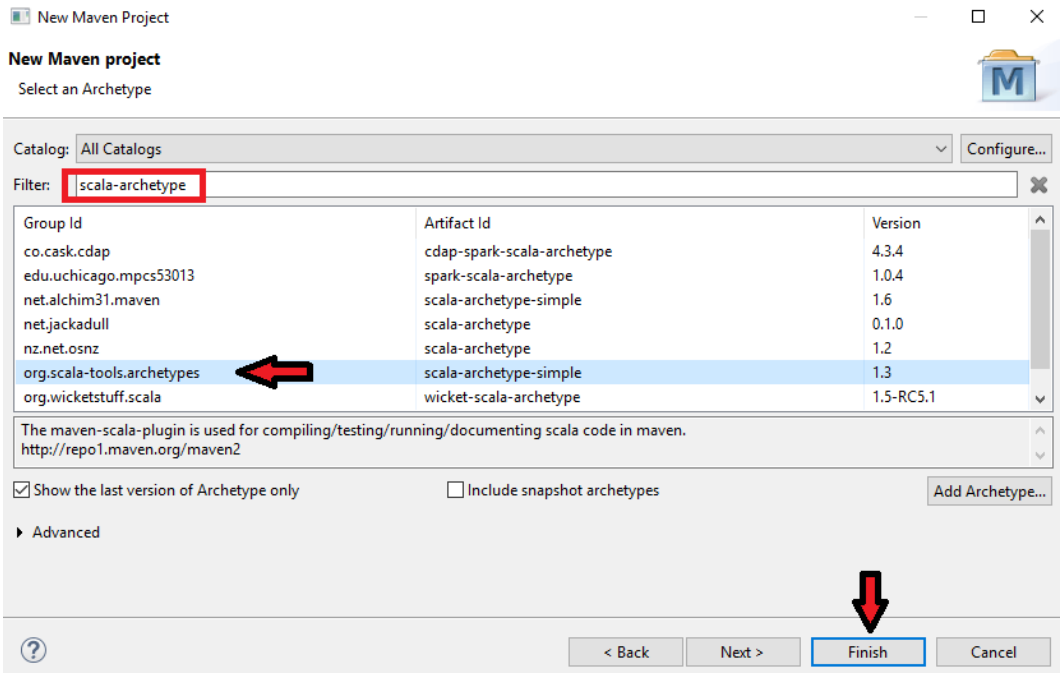


Ilustración 105 - Creación de proyecto Maven

Por último se tiene que añadir dos atributos que sirven para identificar al proyecto Maven y que irán reflejados en el Pom.xml. Estos parámetros que se tienen que indicar son el Group Id y el Artifact Id. El Group Id se le da como valor “TFG-AlbertoMoraga” y en Artifact Id se le indica “KafkaStreams”. Este Artifact Id también será el nombre del proyecto que se podrá ver en el explorador de Eclipse. Una vez que se ha introducido estos parámetros se hace click en “Finish”.

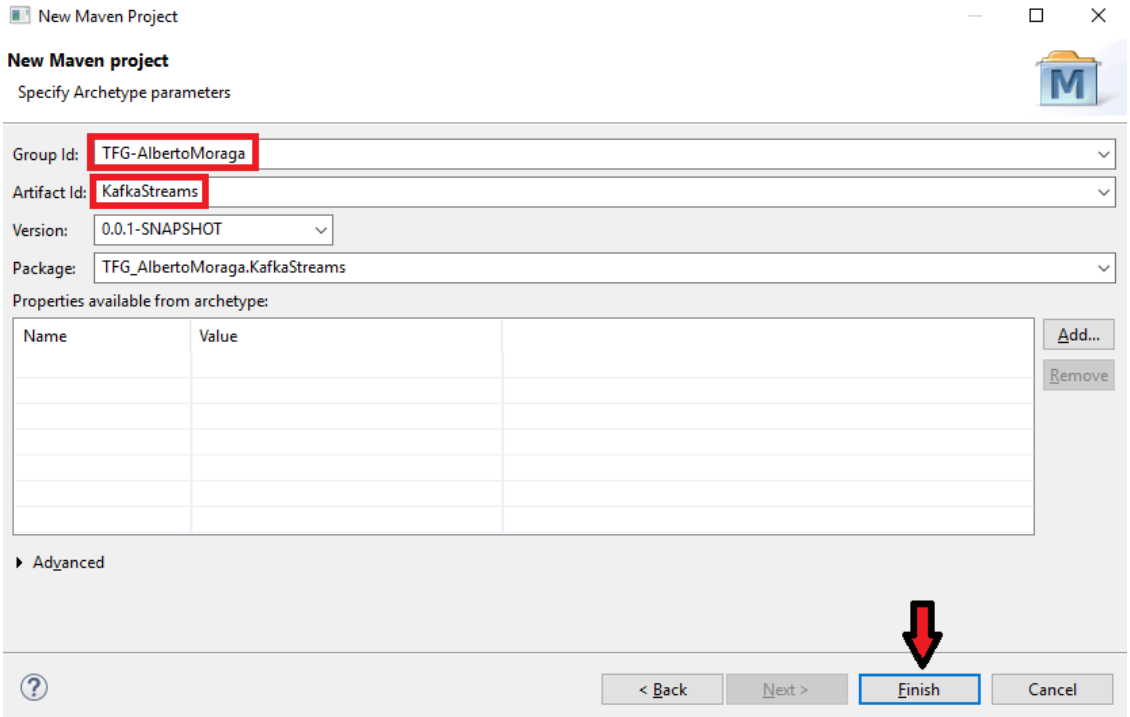


Ilustración 106 - Creación de proyecto Maven

Implementación de Kafka Streams

Una vez que se ha creado nuestro proyecto Maven Scala, esta es la estructura que tenemos:

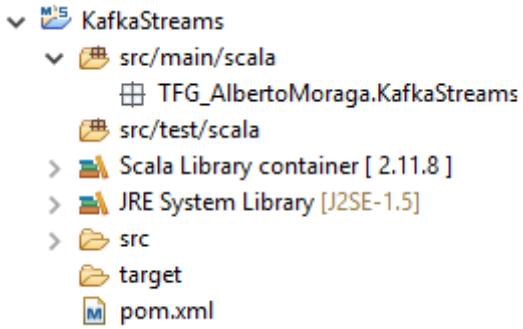


Ilustración 107 - Implementación de Kafka Streams

Como se puede observar todavía no aparecen las dependencias de Maven, porque desde el Pom no se ha indicado ninguna dependencia dentro de la etiqueta "<> dependencies". Este es nuestro Pom inicial:

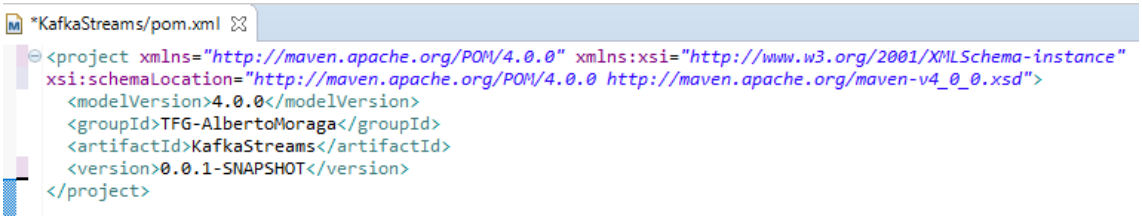


Ilustración 108 - Implementación de Kafka Streams

Hay que dirigirse al paquete "TFG_AlbertoMoraga.KafkaStreams" y dentro de él hay que crear un objeto Scala que se llamará "ConsumerStreams". Para ello se hace click con el botón derecho sobre el paquete New -> Scala Obect. Después únicamente se pone el nombre deseado y se pulsa en "Finish".

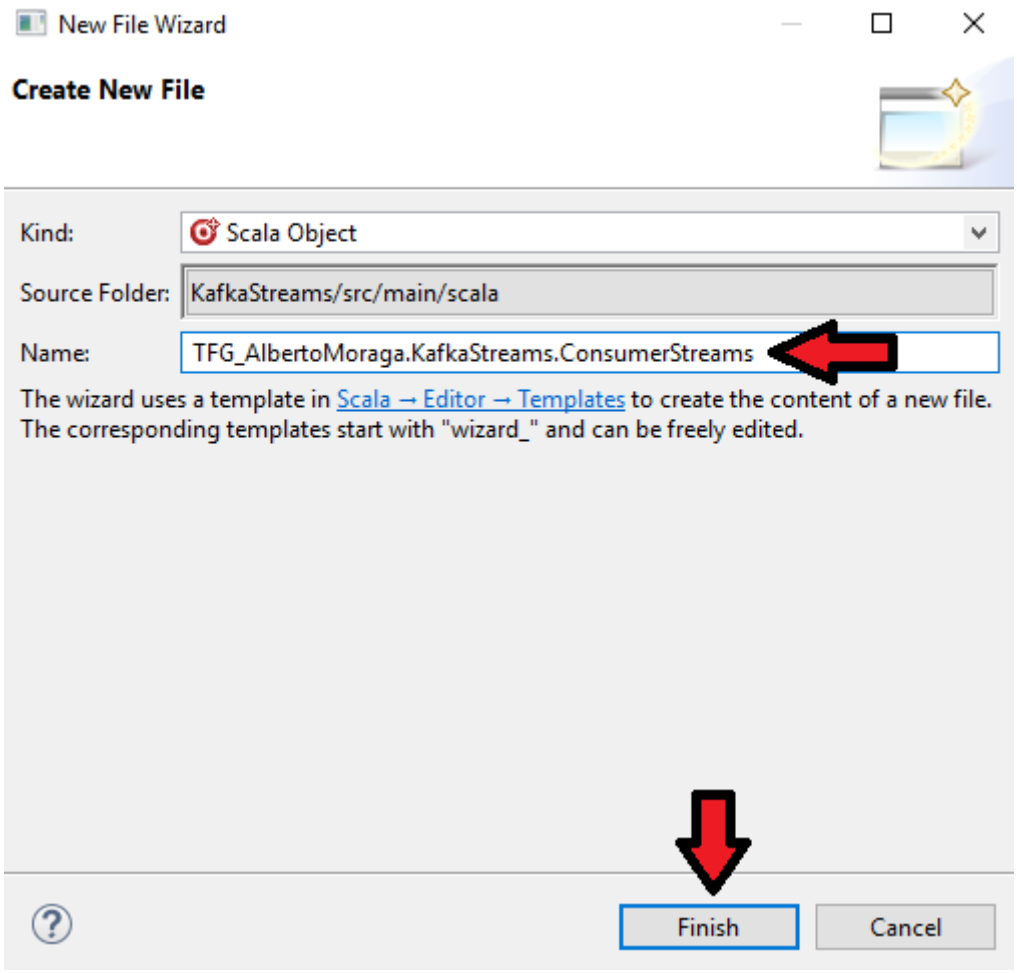


Ilustración 109 - Implementación de Kafka Streams

Una vez se tiene el objeto ConsumerStrems ya se puede comenzar el desarrollo del código para implementar correctamente la herramienta de Kafka Streams y que devuelva en el output un archivo óptimo para realizarle posteriormente el procesamiento Spark necesario. Se va a mostrar el código de la implementación ya completado y posteriormente se realizará una explicación más pormenorizada. Por lo tanto el código con Kafka Streams desarrollado correctamente es el siguiente:

*ConsumerStreams.scala

```scala
package TFG_AlbertoMoraga.KafkaStreams

import org.apache.kafka.streams.processor.WallclockTimestampExtractor
import java.util.Properties
import org.apache.kafka.streams.StreamsConfig
import org.apache.kafka.common.serialization.Serdes
import org.apache.kafka.streams.kstream.KStreamBuilder
import org.apache.kafka.streams.KafkaStreams

object ConsumerStreams {

  def main(args: Array[String]): Unit = {

    val time = new WallclockTimestampExtractor()
    val config = {
      val properties = new Properties()
      properties.put(StreamsConfig.APPLICATION_ID_CONFIG, "Kstreams TFG")
      properties.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092")
      properties.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass)
      properties.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass)
      properties.put(StreamsConfig.DEFAULT_TIMESTAMP_EXTRACTOR_CLASS_CONFIG, time.getClass)
      properties
    }
    val builder = new KStreamBuilder()
    val sourceStream = builder.stream("TopicTweets")
    sourceStream.print()
    sourceStream.writeAsText("C:/Users/amoragaf/Desktop/TFG/Datos para procesamiento Spark/tweets.txt")
    val streams = new KafkaStreams(builder, config)
    streams.start()
  }

}
```

Ilustración 110 - Implementación de Kafka Streams

Dentro del método main se puede diferenciar dos partes principales. La primera de ellas hace referencia a la configuración del Kafka. En la configuración de Kafka que irá dentro de la variable “config” se indica el nombre que se va a dar a nuestra aplicación streaming, la dirección del nodo de Kafka del que tiene que capturar los datos y las clases de serialización / deserialización por defecto para las claves y valores de cada registro. Estas cuatro propiedades son las mínimas necesarias que tienes que programar para cualquier aplicación que su misión sea captar datos en de un topic de Kafka. Sin embargo, estas propiedades en nuestro caso no son suficientes. Cuando empieza a captar los tweets que provienen del topic de Kafka, el programa devuelve un error producido por los Timestamp.

```
Exception in thread "Kstreams TFG-2ee8c14d-949e-4141-917c-c7fbc1ddc70e-StreamThread-1" org.apache.kafka.streams.errors.StreamsException: Input record ConsumerRecord(topic
) has invalid (negative) timestamp. Possibly because a pre-0.10 producer client was used to write this record to Kafka without embedding a timestamp, or because the input
```

Ilustración 111 - Implementación de Kafka Streams

Este error se producía debido al timestamp que lleva asociado cada tweet. Por lo que era necesario añadir a la variable config, la clase que por defecto iba a reconocer el timestamp de los tweets. Para ello se ha tenido que crear un objeto WallclockTimestampExtractor, para posteriormente con un getClass indicárselo en el config.

En la segunda parte del código ya es propiamente de específica de Kafka Strems. En esta parte se añade el KstreamBuilder que tiene un método llamado stream que pasándolo como parámetro el nombre del topic, se encarga de captar los datos de ese topic en concreto. Luego una vez indicado el topic del que tiene que captar los tweets, se indica al programa que vaya mostrando por consola los datos que va cogiendo para comprobar que se está realizando el trabajo correctamente y mucho más importante, crea un archivo en la ruta indicada que será el archivo deseado con los tweets en el formato correcto para el posterior procesamiento en Spark. Para finalizar se inicia el streams pasándole por argumento la configuración y el objeto KstreamBuilder. Como se quiere que se quede escuchando al topic hasta que se pare la ejecución, no hay que incluir el streams.stop().

Una vez que se había desarrollado el main, Eclipse avisaba de algunos fallos en el código. Esto se debía a que había clases que no existían en las librerías iniciales del proyecto. Para tener estas clases y objetos, hay que importar distintas librerías. Para ello se añaden todos los “import” que sean necesarios.

```
import org.apache.kafka.streams.processor.WallclockTimestampExtractor
import java.util.Properties
import org.apache.kafka.streams.StreamsConfig
import org.apache.kafka.common.serialization.Serdes
import org.apache.kafka.streams.kstream.KStreamBuilder
import org.apache.kafka.streams.KafkaStreams
```

Ilustración 112 - Implementación de Kafka Streams

Para importar las librerías necesarias se tendría que buscar y descargar diferentes Jar que coincidiesen en las versiones necesarias de cada una de las librerías y Jar para que no se creasen conflictos entre versiones. Todos estos problemas los vamos a evitar utilizando Maven. Para ello se va a indicar en el pom.xml, las dependencias que se van a necesitar en nuestro proyecto.

*KafkaStreams/pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>TFG-AlbertoMoraga</groupId>
  <artifactId>KafkaStreams</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.apache.kafka/kafka-streams -->
        <dependency>
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka-streams</artifactId>
            <version>0.11.0.0</version>
        </dependency>
  </dependencies>
</project>
```

Ilustración 113 - Implementación de Kafka Streams

Como se puede observar dentro de la etiqueta <dependencies>, se añaden las dependencias necesarias. Para buscar qué dependencia se deben de añadir, hay que dirigirse al repositorio de Maven:

https://mvnrepository.com/artifact/org.apache.maven

Y buscar la herramienta necesaria, en nuestro caso Kafka Streams.

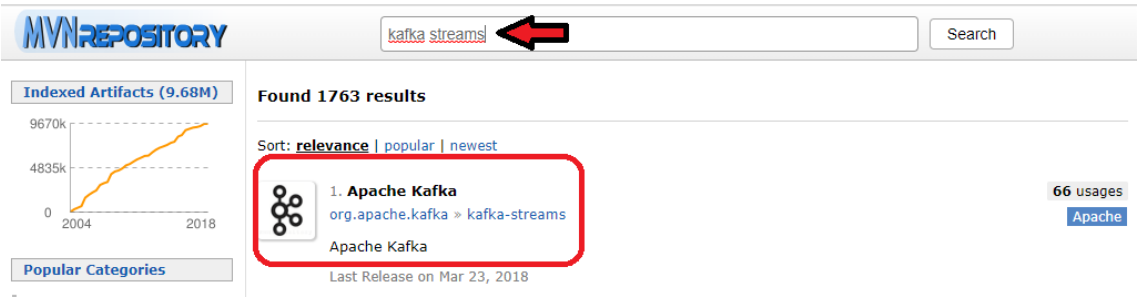


Ilustración 114 - Implementación de Kafka Streams

Una vez seleccionado, únicamente se ve el texto que se tiene que añadir y se añade a nuestro pom. Como para utilizar Kafka Streams necesitas clases de Kafka, Maven ya coordina esas dependencias y añade las librerías necesarias, por lo tanto, la dependencia de Kafka Streams es la única que se tiene que añadir en nuestro pom.

Para concluir se va a comprobar que nuestra aplicación de Kafka Streams, cumple con su cometido y crea correctamente el archivo de los tweets. Se ejecuta el programa que se quedará a la escucha del topic y una vez que se arranque Nifi y Kafka, se comprueban los resultados.

Se ve que la salida por consola es correcta.

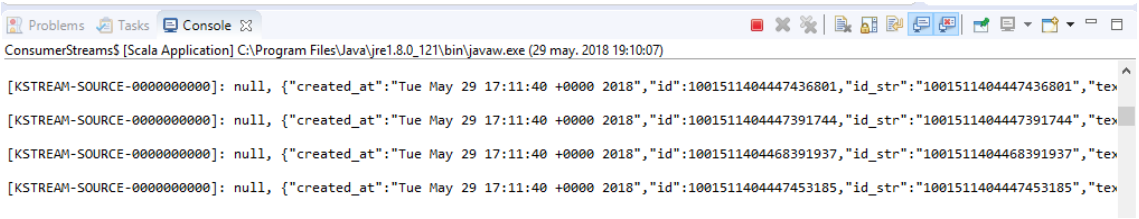


Ilustración 115 - Implementación de Kafka Streams

Y por último, que en la ruta en la que se ha indicado, una vez que hemos parado la aplicación, ha creado un archivo con todos los tweets del topic. Este archivo es al que se va a realizar posteriormente el procesamiento Spark para su futura explotación.

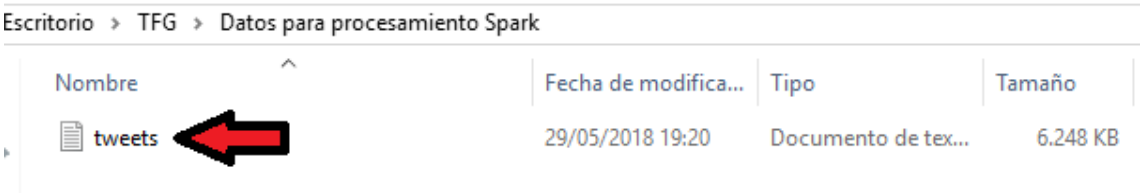


Ilustración 116 - Implementación de Kafka Streams

PROCESAMIENTO Y VISUALIZACION EN APACHE ZEPPELIN

Instalación de la herramienta

Se van a explicar los pasos necesarios para la instalación de Apache Zeppelin en nuestro ordenador. El primer paso es descargarse la herramienta. Se dispone de la versión Apache Zeppelin 0.7.3 que actualmente es la última versión disponible. Para descargarse esta herramienta hay que dirigirse a la pestaña de descargas de la web de Apache Zeppelin:

https://zeppelin.apache.org/download.html

Una vez aquí, se selecciona el tipo de archivo y la versión deseada. Se va a elegir la opción del paquete binario con todos los intérpretes incluidos.

Ilustración 117 - Instalación de la herramienta

Se redirige a la web de Apache Software Foundation. Aquí se ppincha en el enlace de descarga recomendado y acto seguido comienza la descarga de la herramienta.

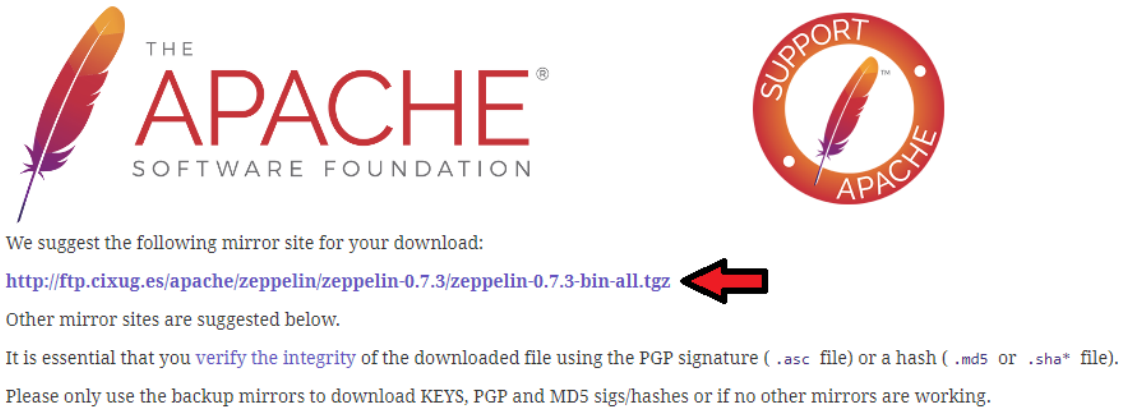


Ilustración 118 - Instalación de la herramienta

Una vez que ya se tiene descargado el archivo comprimido, se debe descomprimir. En nuestro caso, para acceder en el futuro a la herramienta lo más rápido posible ya que cuando abres la consola es la ruta de trabajo que te aparece inicialmente, se ha decidido descomprimirlo en el directorio *C:\Users\amoragaf*

Preparación del archivo

Los RDD (Resilient Distributed Datasets) son las estructuras de datos fundamentales en Spark. Apache Spark, los RDD se pueden crear desde cualquier fuente de datos soportada por Hadoop, incluyendo el sistema de archivos local, HDFS, Cassandra, Hbase, etc.

En nuestro caso se tiene que construir el RDD a partir de un fichero de texto donde están almacenados todos los tweets de nuestro dataset. Para crear un RDD a partir de un archivo de texto, SparkContext tiene un método llamado textFile que pasándole como argumento la ruta del fichero de texto, lo lee como una colección de líneas y crea el RDD.

Nuestro fichero “tweets.txt” contiene todos los tweets que nuestra aplicación de Kafka Streams ha ido capturando del topic de Kafka, que a su vez ha ido almacenando desde la salida de Nifi. Este fichero tiene un problema a la hora de realizarle el procesamiento Spark en Zeppelin. Cuando se realiza el textFile para crear el RDD, Spark se lee línea a línea. El output de Kafka Streams al fichero de texto “tweets.txt” inserta los tweets pero también inserta una línea en blanco entre tweet y tweet. Estos espacios en blanco provocan fallos a la hora de hacer el procesamiento spark del rdd.

Por ello previamente, se va a preparar el archivo antes del procesamiento en zeppelin. Para solucionar la cuestión de la las líneas en blanco, únicamente se tiene que abrir el archivo “tweets.txt” con un editor de texto, en nuestro caso el Notepad++.

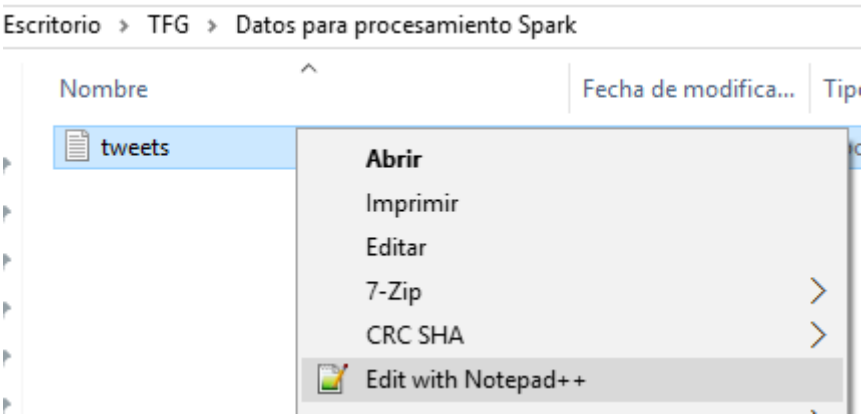


Ilustración 119 - Instalación de la herramienta

Así es como está nuestro archivo “tweets.txt” con los espacios entre los diferentes tweets.

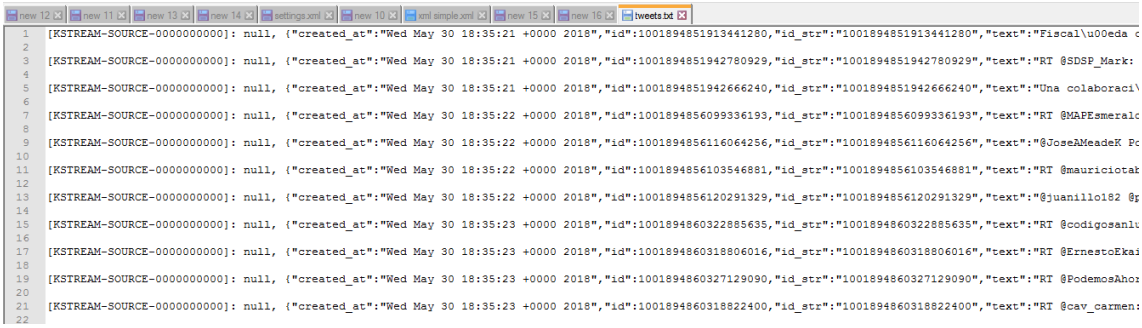


Ilustración 120 - Instalación de la herramienta

Para solucionarlo, hay que dirigirse a Editar -> Operaciones con líneas -> Eliminar líneas vacías

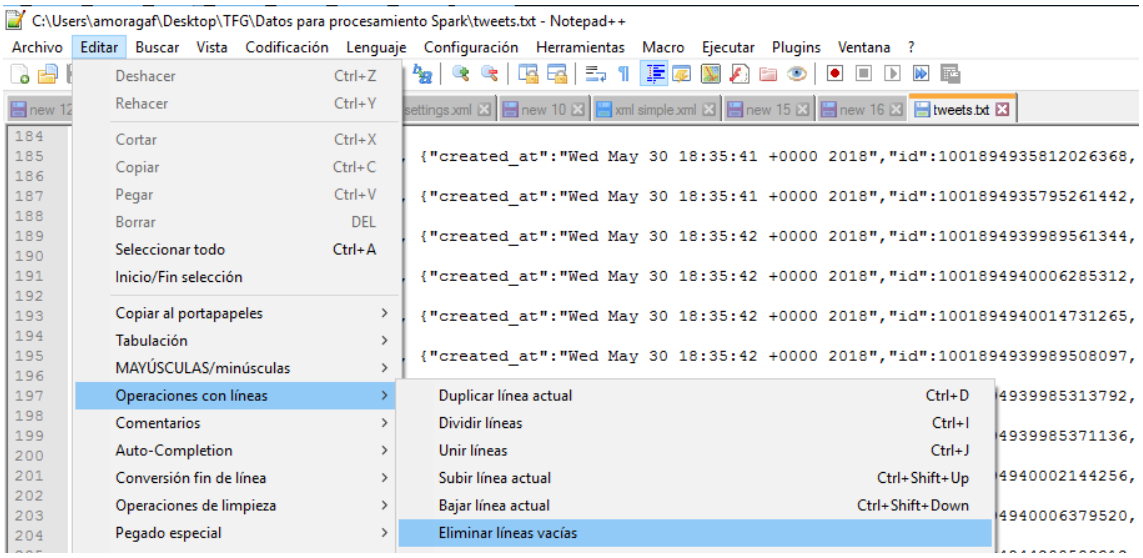


Ilustración 121 - Instalación de la herramienta

Una vez que se ha eliminado las líneas en blanco, cada salto de línea será la delimitación de cada tweet, por lo que a la hora de realizar la creación del rdd con el método textFile del sparkcontext, cada registro del rdd será un tweet.

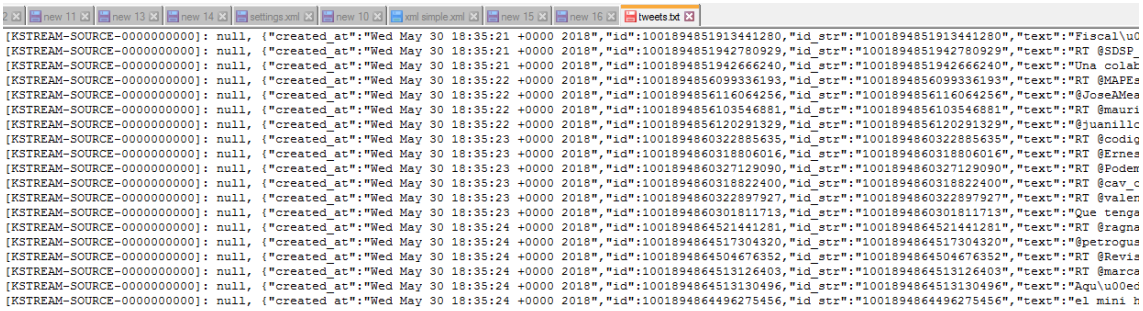


Ilustración 122 - Instalación de la herramienta

Procesamiento Spark

Legado a este punto de nuestra arquitectura Big data, se ha realizado la ingesta de los datos con Nifi, se ha almacenado los datos en un topic de Kafka y se han exportado los datos para su posterior procesamiento y explotación en Apache Zeppelin.

Para realizar el procesamiento de los datos, el primer paso es inicializar la herramienta. Para ello se abre la consola de Windows, se va al directorio donde se descomprimió la herramienta de Zeppelin y en el directorio \bin se ejecuta el archivo “zeppellin.cmd”.

bin\zeppelin.cmd

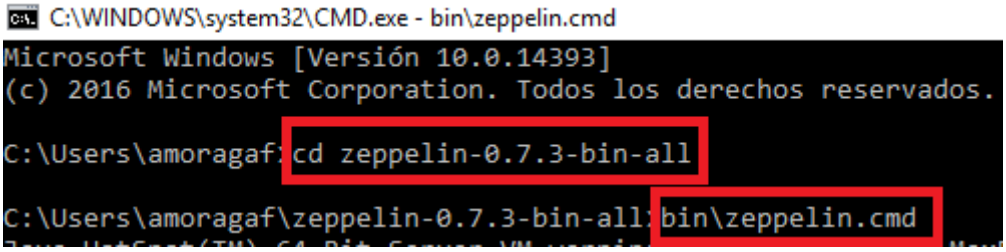


Ilustración 123 - Procesamiento Spark

Pasados unos segundos se ve que la herramienta ya ha arrancado correctamente.

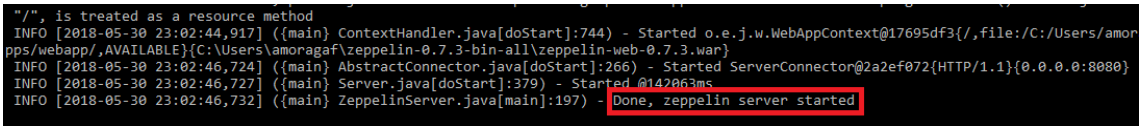


Ilustración 124 - Procesamiento Spark

Para poder trabajar con Zeppelin, se ofrece una interfaz web con la que interactuar con la herramienta. La dirección que se debe introducir en nuestro navegador es: localhost.8080. Con ello se accede a la interfaz principal de la herramienta.

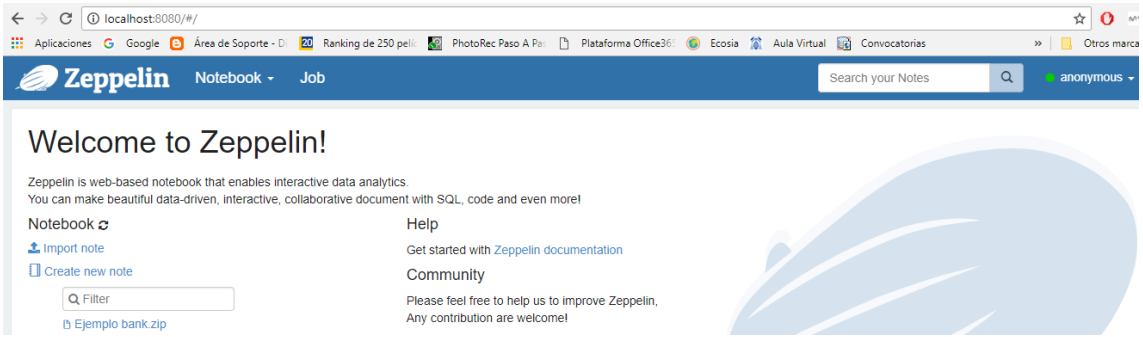


Ilustración 125 - Procesamiento Spark

Toda nuestra implementación en cuanto a la hora de realizar el procesamiento Spark de los datos y de la explotación, se va a realizar en el mismo Note. Esa es una de las principales ventajas de Zeppelin frente a otras herramientas. Dentro de un Note se van a ir añadiendo los llamados paragraphs de código de diferentes lenguajes si fuese necesario, dividirlos visualmente, por ejemplo la zona de procesamiento de los datos por un lado y la explotación, e importarlos y exportarlos. Es decir, que una vez se haya finalizado nuestro note, se puede descargar y cualquier persona se lo podrá importar tal cual para ejecutarlo.

Para crear un note, se hace click en la opción “Create new note”. Pedirá el nombre del note y el intérprete por defecto del note.

Se puede llamar a nuestro note “Twitter procesamiento y visualización” y el intérprete por defecto será spark ya que comenzará con el código de procesamiento en Spark. Posteriormente se pulsa en “Create Note”.

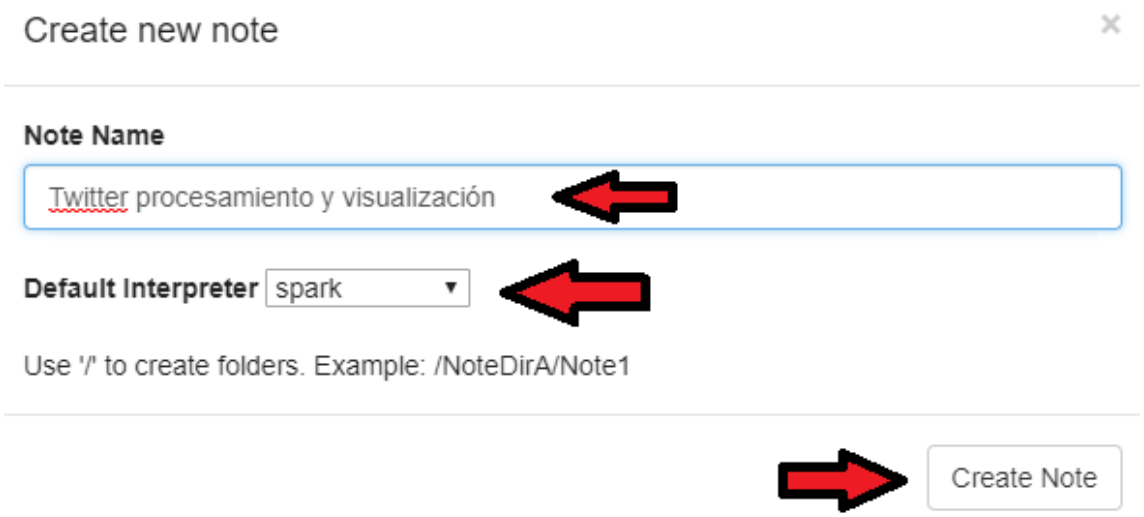


Ilustración 126 - Procesamiento Spark

Vemos que se ha creado nuestro note, con un paragraph vacío para comenzar a desarrollarlo. En el caso que hiciese falta librerías adicionales, se tienen dos opciones. La primera de ellas es en la configuración de los intérpretes de Zeppelin, se tiene la opción de añadir las dependencias y jar que se desee. No es la más conveniente, ya que afectaría a todos los notes de nuestra instalación de Zeppelin y podría generar dependencias no deseadas. La segunda opción es dentro del note en el que se necesiten las librerías, añadir un primer paragraph con las dependencias deseadas. Para indicar que se tratan de dependencias basta con indicar “%dep” al comienzo del paragraph. Esta segunda opción es la que se ha estado utilizando en nuestro trabajo con Zeppelin en las diferentes posibles soluciones, con diferentes librerías, a la hora de abordar nuestro caso de uso. Finalmente al necesitar únicamente los interpretes que ya posee Zeppelin, no ha sido necesario añadir ninguna dependencia ni jar extra.

Llegado este momento se va a comenzar nuestro procesamiento Spark. Al haber puesto como interprete por defecto Spark, no es necesario indicar en el paragraph el intérprete con el que deberá compilar el código que se desarrolle. Se va a mostrar el código del procesamiento necesario para la explotación posterior que se realizará también en Zeppelin, y posteriormente igual que se hizo con nuestra aplicación Kafka Streams, se detallará pormenorizadamente. Cabe recordar que la forma de desarrollar el código Spark en Zeppelin es como lo desarrollaríamos en la Spark-Shell.

El código de procesamiento es el siguiente:

Ilustración 127 - Procesamiento Spark

Para ejecutar el paragrah al completo, hay que dirigirse a las opciones del paragraph situadas arriba a la derecha y pulsar el botón de ejecutar.

Ilustración 128 - Procesamiento Spark

CREACIÓN DEL RDD

La estructura de datos fundamental con la que trabaja Spark son los RDD. Para crear un RDD que contenga los tweets que se quieren procesar, se va a crear el RDD a partir de nuestro archivo en local que ha creado nuestra aplicación de Kafka Streams. Para ello se va a utilizar la función textFile que ofrece el SparkContext. Hay que recordar que el sparContext en Zeppelin está creado por defecto dándole de nombre “sc”. Se le pasa a la función, como parámetro la ruta del directorio en la que nuestra aplicación de Kafka Streams ha exportado los datos de los tweets.

```
val tweetsRDD = sc.textFile("C:/Users/amoragaf/Desktop/TFG/Datos para procesamiento Spark/tweets.txt")
```

COMPROBACIÓN DE TWEETSRDD

Para comprobar que se ha creado bien el RDD, se va a sacar por pantalla. Para ello se utiliza el método foreach del rdd, que realiza una función a cada registro que en este caso será la de println. Este es el método más recomendado para pintar un rdd por pantalla. Con el método take se coge únicamente el primer registro.

```
tweetsRDD.take(1).foreach(println)
```

SPLIT A TODOS LOS REGISTROS

En estos momentos se dispone de un RDD con multitud de registros en los cuales cada registro representa a la información de un tweet, pero lo que a nosotros nos interesa son los diferentes campos de esos tweets. Para ello se va a realizar lo que se denomina transformación en Spark. Las transformaciones aplican una operación a cada registro del RDD, creando un nuevo RDD resultante. La transformación que se realiza es un map en el que la función split nos va a dividir cada registro del RDD en diferentes campos indicándole el delimitador por el que están separados esos campos. En nuestro caso el delimitador es ´”´. Este nuevo RDD lo llamamos splitRDD.

```
val splitRDD = tweetsRDD.map(x => x.split('"'))
```

REGISTROS CON LOS CAMPOS QUE NOS INTERESAN

Se va a realizar otra transformación. Ahora que se tienen los registros del RDD separados por los diferentes parámetros que los forman, se va a crear un nuevo RDD con únicamente los campos que interesan de cara a la visualización. Estos campos son el “id_str” que es el identificador inequívoco del tweet, el “created_at” que es la fecha exacta de la creación del tweet y el “text” que es el propio texto del tweet. A este nuevo RDD le hemos llamado atributosRDD.

```
val atributosRDD = splitRDD.map(x => (x(9),x(3),x(13)))
```

COMPROBACIÓN DE ATRIBUTOSRDD

Al igual que se hizo anteriormente, para comprobar que hasta este punto las transformaciones que se están realizando sobre los RDD están siendo las correctas, se imprime por pantalla uno de sus registros.

```
atributosRDD.take(1).foreach(println)
```

CONVERSION DE RDD A DATAFRAME

En este paso se va a convertir nuestro RDD en un Dataframe. Un Dataframe es una colección de datos distribuida, la cual se organiza por columnas. Un Dataframe se puede construir a partir de tablas Hive, archivos de datos estructurados, bases de datos externas o de RDDs ya existentes.

```
val atributosDF = atributosRDD.toDF()
```

CREACIÓN DE LA TEMPTABLE

Se ha creado un Dataframe del RDD que se tenía para poder utilizar el método registerTempTable, que creará una tabla con todos nuestros datos, que permitirá realizar consultas sql sobre dicha tabla. A la tabla se ha denominado Tweets.

```
atributosDF.registerTempTable("Tweets")
```

Visualización de los datos / explotación

Llegados a este punto de la arquitectura, el recorrido del dato ha sido el correcto a lo largo de todo nuestro entorno Big data y se le han aplicado todos los procesos necesarios. Llega el momento de poder visualizar los datos, realizando la explotación deseada de ellos y siendo capaces de sacar información y conclusiones de los datos ingestados.

La visualización de los datos al igual que el procesamiento Spark se va a realizar en la herramienta Apache Zeppelin. Se trata de uno de los puntos fuertes de la herramienta, ya que en el mismo note en el que hemos desarrollado el paragraph donde realizábamos el procesamiento Spark de los datos, es decir, en el note "Twitter procesamiento y visualización" se va a ir añadiendo paragraphs en los que se definirán diferentes funciones, realizarán consultas y se irá explotando la información con diferentes tipos de formas visuales para analizarla.

A continuación, se va a mostrar todos los pasos que se han seguido para realizar una visualización y explotación de los datos completa ofreciéndonos el máximo de información representativa.

MOSTRAR LAS TABLAS CREADAS

Esta consulta va a servir para comprobar que se ha construido la tabla a partir de nuestro dataframe. Nos mostrará todas las tablas creadas. Como se trata de una consulta sql y el intérprete por defecto es el de Spark, se tiene que comenzar el código indicándole el intérprete con el cuál lo tiene que ejecutar, es decir, "%sql".

```
%sql

SHOW TABLES
```

Ilustración 129 - MOSTRAR LAS TABLAS CREADAS

Como resultado devuelve la tabla que se ha creado creado a partir del Dataframe:

tableName	isTemporary
tweets	true

Ilustración 130 - MOSTRAR LAS TABLAS CREADAS

INFORMACIÓN DE UNA TABLA EN CONCRETO

En esta consulta se va a seleccionar la tabla que se ha creado anteriormente a partir del Dataframe y devolverá la información de dicha tabla. Cada uno de los campos que se seleccionan gracias al delimitador con la función Split, que se utilizó anteriormente en el procesamiento en Spark, serán cada una de las columnas de nuestra tabla.

```
%sql

DESCRIBE tweets
```

Ilustración 131 - INFORMACIÓN DE UNA TABLA EN CONCRETO

col_name	data_type	comment
_1	string	null
_2	string	null
_3	string	null

Ilustración 132 - INFORMACIÓN DE UNA TABLA EN CONCRETO

CONSULTAR LOS DATOS DE UNA DE LAS COLUMNAS DE LA TABLA

Puede que interese consultar únicamente el texto o cuerpo de los tweets, obviando su fecha de creación y su identificador inequívoco. Para ello se va a realizar un Select de un único campo de la columna que corresponderá al texto de los tweets.

Para que la explotación por parte del usuario sea lo más eficiente y comprensiva, se cambia el nombre de la columna que es "_3" y es poco representativo, por "text".

```
%sql

SELECT _3 AS text

FROM tweets
```

Ilustración 133 - CONSULTAR LOS DATOS DE UNA DE LAS COLUMNAS DE LA TABLA

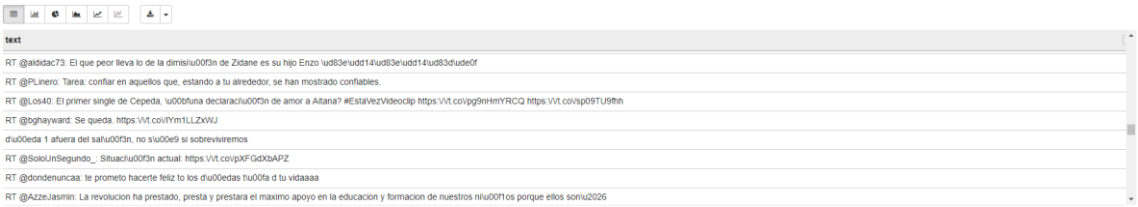


Ilustración 134 - CONSULTAR LOS DATOS DE UNA DE LAS COLUMNAS DE LA TABLA

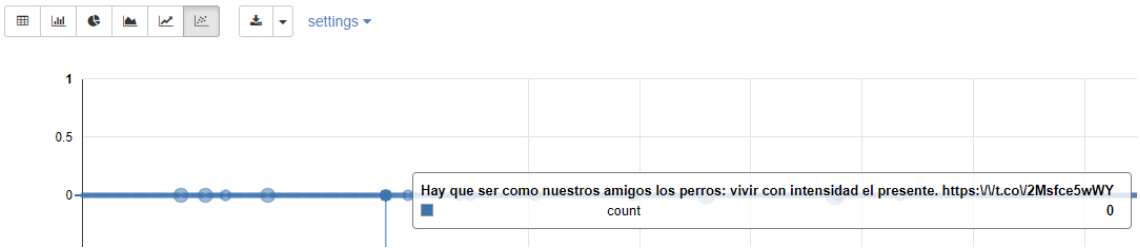


Ilustración 135 - CONSULTAR LOS DATOS DE UNA DE LAS COLUMNAS DE LA TABLA

CONSULTAR TODOS LOS DATOS DE LA TABLA

Esta consulta permite visualizar todos los datos al completo. Cada fila de la tabla corresponderá con cada registro que se ha almacenado. Se podrá visualizar todos los tweets almacenados, con su correspondiente id y su fecha de creación.

```
%sql

SELECT *

FROM tweets
```

Ilustración 136 - CONSULTAR LOS DATOS DE UNA DE LAS COLUMNAS DE LA TABLA

_1	_2	_3
1002512085702541314	Fri Jun 01 11:28:01 +0000 2018	Quedan 183 d\u00edas y es lo que me preocupa \ud83d\ude0c
1002512085715247106	Fri Jun 01 11:28:01 +0000 2018	RT @MashiRafael: Un abrazo fraterno y solidario a todos nuestros compatriotas en Italia, especialmente a la familia de Paulina Calahorrano.\u2026
1002512085710987264	Fri Jun 01 11:28:01 +0000 2018	RT @_SantosTrinidad: Karam Majdi solicitante de asilo de 19 a\u00f1os, que intent\u00f3 entrar en Gran Breta\u00f1a mintiendo, al decir que era un ni\u00f1o hu\u2026
1002512085715169280	Fri Jun 01 11:28:01 +0000 2018	RT @alejandro111169: @patriciaanacg1 @CNNChile @sergioulloa1 Insisto esta es tan pero tan https:\/\/t.co\/RVe5o8I4Ca
1002512089896845312	Fri Jun 01 11:28:02 +0000 2018	RT @carloshuitron3: Mitad de a\u00f1o y no he hecho la mitad de las cosas que pensaba hacer este a\u00f1o
1002512089880186880	Fri Jun 01 11:28:02 +0000 2018	@Jona3March @gaceta_es @maria8769 Verg\u00fcenza es dar una paliza a unos GC solo por serlo. El unico regimen fascista e\u2026 https:\/\/t.co\/DAerW4uvOv
1002512094108028930	Fri Jun 01 11:28:03 +0000 2018	RT @PirataGH: Sois unos mentirosos y no cumplis vuestra palabra @Supervivientes:\n1. Dijisteis que a la pr\u00f3xima ser\u00eda nominado, no dijisteis\u2026
1002512094095429632	Fri Jun 01 11:28:03 +0000 2018	@antonia_frank17 Dici???\ud83d\ude07

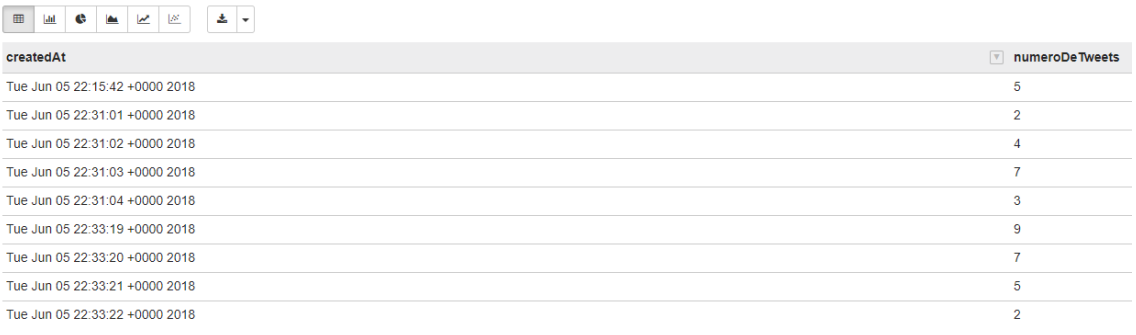
Ilustración 137 - CONSULTAR LOS DATOS DE UNA DE LAS COLUMNAS DE LA TABLA

OBTENER NÚMERO DE TWEETS POR SEGUNDO

Como usuarios, pueden interesar diferentes métricas de nuestros datos. En esta consulta se va a ofrecer al usuario la posibilidad de consultar una de esas métricas. Se va a calcular el número de tweets que se disponen de cada instante de tiempo. Para ello se realiza el cálculo de las veces que se repite un mismo instante de tiempo, se agrupan por el momento de creación del tweet y además para que sea clarificador visualmente se ha ordenado temporalmente.

```
%sql

SELECT _2 AS createdAt, count(1) AS numeroDeTweets

FROM tweets

GROUP BY _2

ORDER BY _2
```

Ilustración 138 - OBTENER NÚMERO DE TWEETS POR SEGUNDO



createdAt	numeroDeTweets
Tue Jun 05 22:15:42 +0000 2018	5
Tue Jun 05 22:31:01 +0000 2018	2
Tue Jun 05 22:31:02 +0000 2018	4
Tue Jun 05 22:31:03 +0000 2018	7
Tue Jun 05 22:31:04 +0000 2018	3
Tue Jun 05 22:33:19 +0000 2018	9
Tue Jun 05 22:33:20 +0000 2018	7
Tue Jun 05 22:33:21 +0000 2018	5
Tue Jun 05 22:33:22 +0000 2018	2

Ilustración 139 - OBTENER NÚMERO DE TWEETS POR SEGUNDO

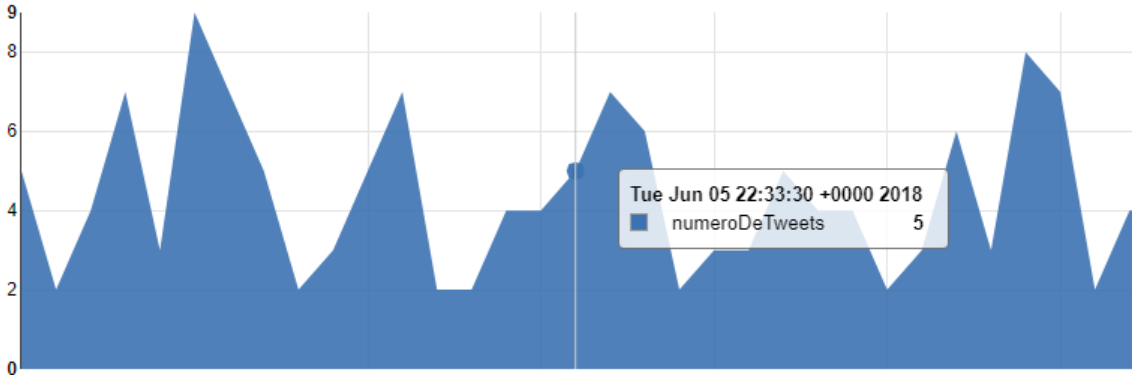


Ilustración 140 - OBTENER NÚMERO DE TWEETS POR SEGUNDO

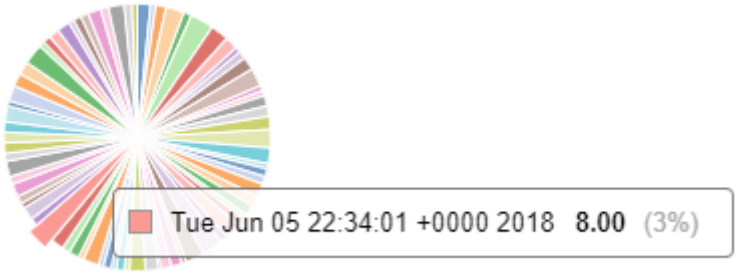


Ilustración 141 - OBTENER NÚMERO DE TWEETS POR SEGUNDO

MOSTRAR ÚNICAMENTE LOS RETWEETS

Cuando se está consultando el timeline de Twitter, pueden aparecer dos tipos de contenidos que publican tus seguidores. Por un lado están los tweets que crean los usuarios y por otro lado están los retweets que hacen de otros usuarios que han creado previamente. En esta consulta se va a mostrar únicamente aquellos que son retweets. La API de twitter representa los retweets inidcando “RT @nombreDelUsuario” al comienzo del campo “text” del tweet. Esa es la forma en la que los identificaremos en nuestra consulta sql.

```
%sql
SELECT _3 AS Retweets

FROM tweets

WHERE _3 LIKE 'RT @%'
```

Ilustración 142 - MOSTRAR ÚNICAMENTE LOS RETWEETS

Retweets
RT @MashiRafael: Un abrazo fraterno y solidario a todos nuestros compatriotas en Italia, especialmente a la familia de Paulina Calahorrano,\u2026
RT @_SantosTrinidad: Karam Majdi solicitante de asilo de 19 a\u00f1os, que intent\u00f3 entrar en Gran Breta\u00f1a mintiendo, al decir que era un ni\u00f1o hu\u2026
RT @alejandro111169: @patriciaanacg1 @CNNChile @sergioulloa1 Insisto esta es tan pero tan https:\/\/t.co\/RVe5o8l4Oa
RT @carloshuitron3: Mitad de a\u00f1o y no he hecho la mitad de las cosas que pensaba hacer este a\u00f1o
RT @PirataGH: Sois unos mentirosos y no cumplis vuestra palabra @Supervivientes:\n1. Dijisteis que a la pr\u00f3xima ser\u00eda nominado, no dijisteis\u2026
RT @alonso_dm: Rajoy acaba de salir de una guarder\u00eda y les ha dicho a los ni\u00f1os: ah\u00ed os las apa\u00f1\u00e9is. Se van a destrozar de tal manera entre\u2026
RT @IlGrossoChelo: El cacerolazo de @Solanopo \nMenos gente q un recital de Locomia gratis.\n#DejaDePasarVerguenza \n#Payaso https:\/\/t.co\/7fkg\u2026
RT @RubenSanchezTW: Albert Rivera ha votado NO a echar al partido m\u00e1s corrupto de Europa. A partir de ma\u00f1ana dir\u00e1 que gracias a Ciudadanos\u2026

Ilustración 143 - MOSTRAR ÚNICAMENTE LOS RETWEETS

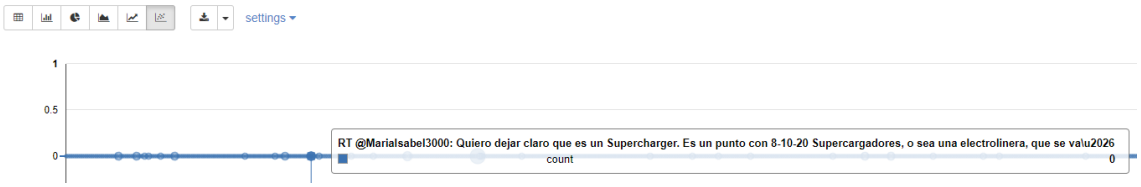


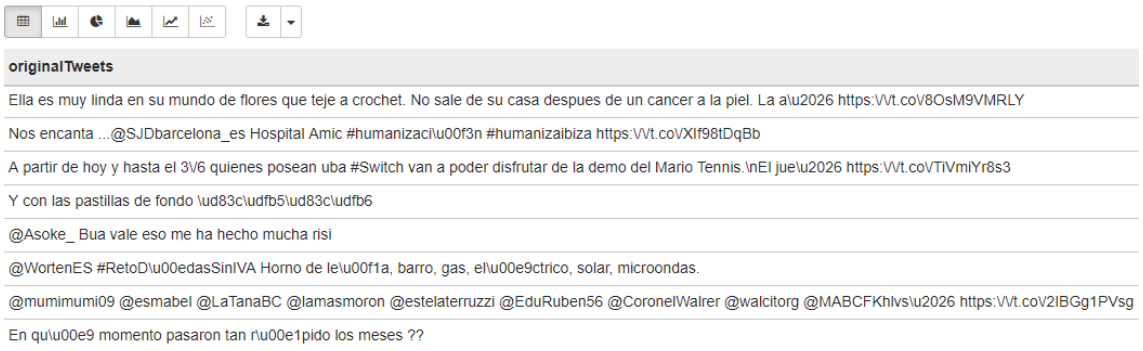
Ilustración 144 - MOSTRAR ÚNICAMENTE LOS RETWEETS

MOSTRAR LOS TWEETS DE NUEVA CREACIÓN

Tal y como comentábamos en la anterior consulta existen dos tipo de tweets que te aparecen en tu timeline. Los tweets de nueva creación son aquellos que los usuarios crean. Pueden sernos muy interesantes a la hora de analizar, porque son los que están añadiendo contenido nuevo en la red social.

```
%sql
SELECT _3 AS Retweets

FROM tweets

WHERE _3  NOT LIKE 'RT @%'
```

Ilustración 145 - MOSTRAR LOS TWEETS DE NUEVA CREACIÓN



originalTweets
Ella es muy linda en su mundo de flores que teje a crochet. No sale de su casa despues de un cancer a la piel. La a\u2026 https:\/\/t.co\/8OsM9VMRLY
Nos encanta ...@SJDbarcelona_es Hospital Amic #humanizaci\u00f3n #humanizaibiza https:\/\/t.co\/XIf98tDqBb
A partir de hoy y hasta el 3\/6 quienes posean uba #Switch van a poder disfrutar de la demo del Mario Tennis.\nEl jue\u2026 https:\/\/t.co\/TiVmiYr8s3
Y con las pastillas de fondo \ud83c\udfb5\ud83c\udfb6
@Asoke_ Bua vale eso me ha hecho mucha risi
@WortenES #RetoD\u00edasSinIVA Horno de le\u00f1a, barro, gas, el\u00e9ctrico, solar, microondas.
@mumimumi09 @esmabel @LaTanaBC @lamasmoron @estelaterruzzi @EduRuben56 @CoronelWalrer @walcitorg @MABCFKhlvs\u2026 https:\/\/t.co\/2IBGg1PVsg
En qu\u00e9 momento pasaron tan r\u00e1pido los meses ??

Ilustración 146 - MOSTRAR LOS TWEETS DE NUEVA CREACIÓN

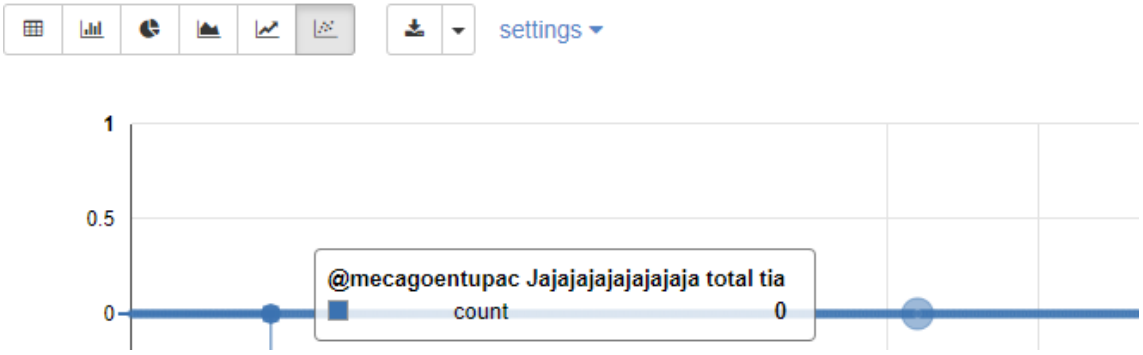


Ilustración 147 - MOSTRAR LOS TWEETS DE NUEVA CREACIÓN

CREAR FUNCIÓN CONTENIDO NUEVO VS RETWEETS

Hasta el momento se han ido haciendo consultas sql sobre nuestra tabla temporal que se creó en Spark, donde se encuentran los datos que interesan de los tweets que se ingestan. Sin embargo Zeppelin permite en el mismo note donde se está desarrollando todos los paragraphs, añadir paragraphs en los que crear nuestras propias funciones en Scala que se podrán utilizar en la explotación de nuestros datos.

Se va a crear una función en Scala para poder realizar una comparación entre dos tipos de tweets. Para ello no hay que indicar el intérprete como en otros paragraphs, ya se ha definido el de spark y es el lenguaje con el que compila spark. Aprovechando la forma en la que la API de Twitter identifica los retweets y los tweets originales se podrán diferenciar.

```
def tipoDeTweet(s:String) : String = {

   if(s.startsWith("RT @"))
    "RT"
    else
    "Original tweet"

}


sqlc.udf.register("tipoDeTweet", tipoDeTweet _)
```

Ilustración 148 - CREAR FUNCIÓN CONTENIDO NUEVO VS RETWEETS

MOSTRAR CONTENIDO NUEVOS VS RETWEETS

Una vez que ya está definida la función que devuelve el tipo de tweet del que se trata cada registro, se está en disposición de realizar una consulta sql utilizando la función que se ha creado. Además para poder hacer una comparación entre ambos tipos, se utilizará la función “count”.

```
%sql

SELECT  tipoDeTweet(_3) AS tipo, count(1) AS numberOfTweets

FROM tweets

GROUP BY tipoDeTweet(_3)
```

Ilustración 149 - MOSTRAR CONTENIDO NUEVOS VS RETWEETS

tipo	numberOfTweets
Original tweet	1280
RT	2164

Ilustración 150 - MOSTRAR CONTENIDO NUEVOS VS RETWEETS

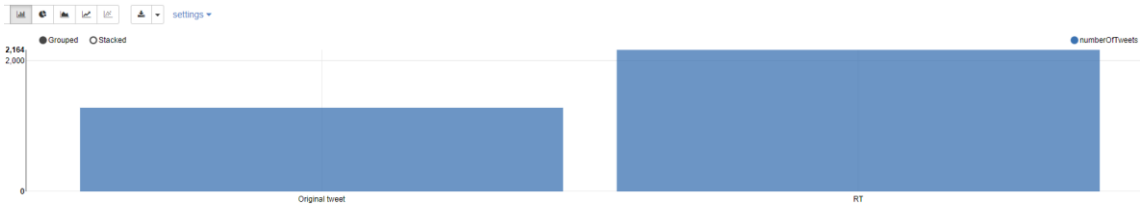


Ilustración 151 - MOSTRAR CONTENIDO NUEVOS VS RETWEETS

Ilustración 152 - MOSTRAR CONTENIDO NUEVOS VS RETWEETS

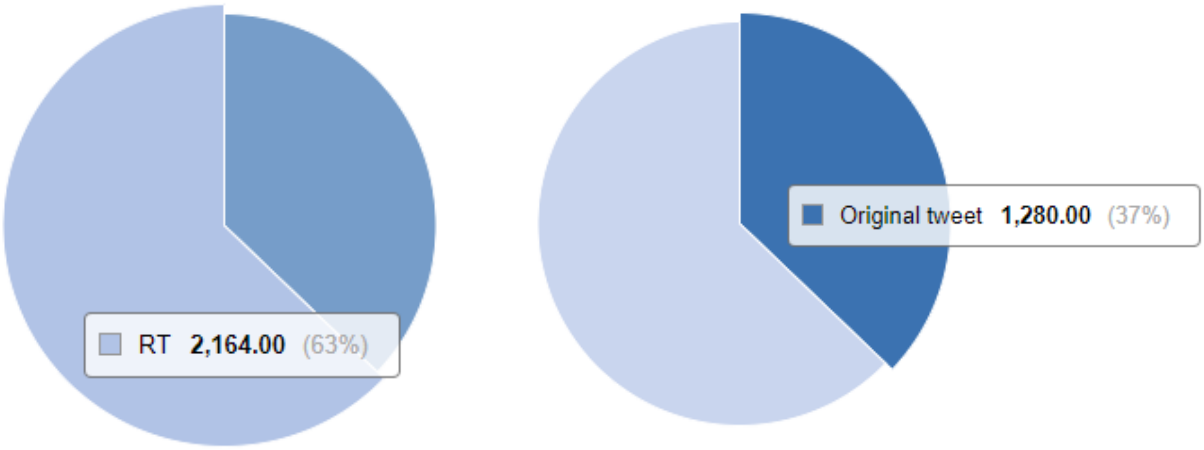


Ilustración 153 - MOSTRAR CONTENIDO

CREAR FUNCIÓN CONTAR PALABRAS DE UN TWEET

A la hora de poder realizar métricas más profundas de los tweets, se siguen creando nuestras propias funciones. En este caso se va a desarrollar en Python y tendrá como misión contar las palabras que el usuario ha añadido en el tweet.

Para indicar a Zeppelin que el intérprete con el que tendrá que ejecutar es Python se debe añadir al inicio del paragraph "%pyspark".

```
%pyspark

def wordcount(a):
  return len(re.split("\W+",a))

sqlContext.registerFunction("wordcount", wordcount)
```

Ilustración 154 - CREAR FUNCIÓN CONTAR PALABRAS DE UN TWEET

UTILIZACIÓN FUNCIÓN CONTAR PALABRAS

Una vez que ya está definida la función que devuelve el número de palabras de cada tweet, está en disposición de realizar una consulta sql utilizando la función que se ha creado.

Una de las consultas que se podría realizar, es mostrar información más detallada de la consulta de los tipos de tweets, mostrando en una nueva columna, a parte del tipo de tweet, el número de palabras que lo forman.

```
%sql

SELECT _1 AS idTweet, tipoDeTweet(_3) AS tipo, wordcount(_3) AS numPalabras, _3 AS texto

FROM tweets
```

Ilustración 155 - UTILIZACIÓN FUNCIÓN CONTAR PALABRAS

idTweet	tipo	numPalabras	texto
1002512085702541314	Original tweet	12	Quedan 183 d\u00edas y es lo que me preocupa \ud83d\ude0c
1002512085715247106	RT	21	RT @MashiRafael: Un abrazo fraterno y solidario a todos nuestros compatriotas en Italia, especialmente a la familia de Paulina Calahorrano,\u2026
1002512085710987264	RT	29	RT @_SantosTrinidad: Karam Majdi solicitante de asilo de 19 a\u00f1os, que intent\u00f3 entrar en Gran Breta\u00f1a mintiendo, al decir que era un ni\u00f1o hu\u2026
1002512085715169280	RT	15	RT @alejandro111169: @patriciaanacg1 @CNNChile @sergioulloa1 Insisto esta es tan pero tan https:\/\/t.co\/RVe5o8l4Oa
1002512089896845312	RT	21	RT @carloshuitron3: Mitad de a\u00f1o y no he hecho la mitad de las cosas que pensaba hacer este a\u00f1o
1002512089880186880	Original tweet	26	@Jona3March @gaceta_es @maria8769 Verg\u00fcenza es dar una paliza a unos GC solo por serlo. El unico regimen fascista e\u2026 https:\/\/t.co\/DAerW4uvOv
1002512094108028930	RT	24	RT @PirataGH: Sois unos mentirosos y no cumplis vuestra palabra @Supervivientes:\n1. Dijisteis que a la pr\u00f3xima ser\u00eda nominado, no dijisteis\u2026
1002512094095429632	Original tweet	5	@antonia_frank17 Dici???\ud83d\ude07

Ilustración 156 - UTILIZACIÓN FUNCIÓN CONTAR PALABRAS

CREAR FUNCIÓN ANÁLISIS DE SENTIMIENTO

En la campaña para las elecciones presidenciales de Estados Unidos de 2016, el análisis de las redes sociales jugó un papel crucial a la hora de encarar y diseñar las estrategias a seguir por los candidatos. Según iban avanzando en la campaña iban conociendo con precisión cuales eran las reacciones de los votantes.

En nuestro caso se ha creado una función de análisis de sentimiento similar a las utilizadas en aquella campaña, donde se van a analizar el sentimiento de los tweets. Esta función va a analizar las diferentes palabras de los tweets y los va a clasificar según sentimiento en positivo, negativo o neutral. Puntualizar que se ha decidido realizar para tweets en inglés ya que es más fácil a la hora de buscar las palabras que indicarán un tipo de sentimiento u otro ya que no existen tantas formas verbales como en español.

```
def sentimiento(s:String) : String = {
    val positivo = Array("like", "love", "good", "great", "happy", "cool", "the", "one", "that")
    val negativo = Array("hate", "bad", "stupid", "is")

    var total = 0;

    val words = s.split(" ")
    positivo.foreach(p =>
        words.foreach(w =>
            if(p==w) total = total+1
        )
    )

    negativo.foreach(p=>
        words.foreach(w=>
            if(p==w) total = total-1
        )
    )
    if(total > 0)
        "positivo"
    else if(total < 0)
        "negativo"
    else
        "neutral"
}

sqlc.udf.register("sentimiento", sentimiento _)
```

Ilustración 157 - CREAR FUNCIÓN ANÁLISIS DE SENTIMIENTO

MOSTRAR ANÁLISIS DE LOS SENTIMIENTOS

Para realizar esta consulta únicamente comentar que no se ha utilizado el Dataset que se ha venido utilizando para el resto de consultas previas, si no que se ha hecho una ingesta de datos exclusiva para esta consulta. Para ello a la hora de realizar la ingesta de datos en Apache Nifi se ha seleccionado como idioma el inglés. El resto del recorrido del dato a través de la arquitectura Big Data y los procesos realizados son exactamente los mismos.

```
%sql

SELECT  sentimiento(_3) AS sentimiento, count(1) AS numTweets

FROM tweets

GROUP BY sentimiento(_3)
```

Ilustración 158 - MOSTRAR ANÁLISIS DE LOS SENTIMIENTOS

sentimiento	numTweets
neutral	1685
negativo	198
positivo	887

Ilustración 159 - MOSTRAR ANÁLISIS DE LOS SENTIMIENTOS

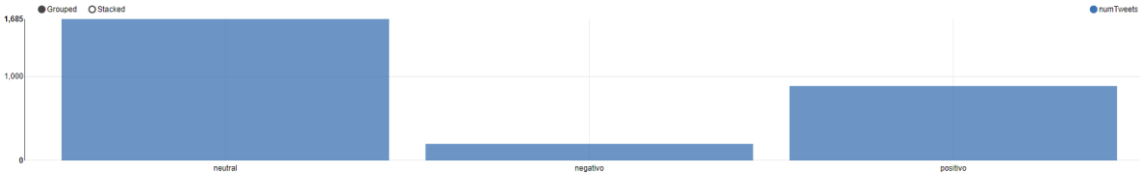


Ilustración 160 - MOSTRAR ANÁLISIS DE LOS SENTIMIENTOS

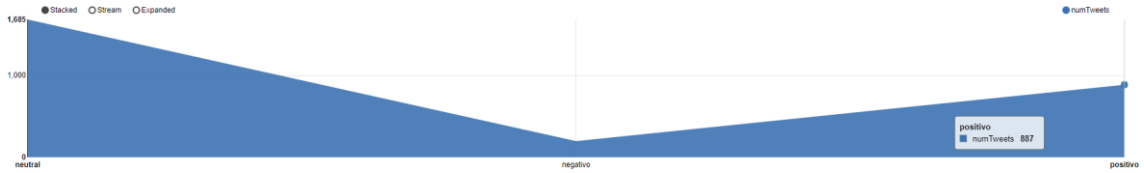


Ilustración 161 - MOSTRAR ANÁLISIS DE LOS SENTIMIENTOS

Ilustración 162 - MOSTRAR ANÁLISIS DE LOS SENTIMIENTOS

CREAR FUNCIÓN PRIMERA O SEGUNDA PERSONA

Siguiendo con las funciones de análisis del texto de los tweets, utilizando el esquema de la función creada para los sentimientos, se ha desarrollado una función para determinar la persona que utiliza el usuario para expresarse a la hora de crear el tweet. Para ello utilizando los pronombres y otras palabras identificativas, se ha podido analizar el texto de los tweets.

Esta función determinará de qué manera se expresan los usuarios de Twitter y se podrá acceder al tipo que se desea analizar.

```
def persona(s:String) : String = {
    val primera = Array("yo", "he", "me", "hemos", "nosotros", "mi", "conmigo", "nos", "nosotras")
    val segunda = Array("tu", "ti", "te","contigo", "vosotras", "vos", "os", "vosotros", "has", "habeis")

    var st = 0;

    val words = s.split(" ")
    primera.foreach(p =>
        words.foreach(w =>
            if(p==w) st = st+1
        )
    )

    segunda.foreach(p=>
        words.foreach(w=>
            if(p==w) st = st-1
        )
    )
    if(st>0)
        "primera"
    else if(st<0)
        "segunda"
    else
        "otro"
}

sqlc.udf.register("persona", persona _)
```

Ilustración 162 - CREAR FUNCIÓN PRIMERA O SEGUNDA PERSONA

UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

Una vez que se tiene implementada la función en Scala, se puede utilizar en cualquiera de nuestras consultas. En esta primera consulta se va a obtener el número de tweets de cada tipo de persona que se ha identificado (utilizando para ello la sentencia Group by), obteniendo además el porcentaje de cada tipo sobre el total en diferentes modos de visualización.

```
%sql

SELECT  sentimiento(_3) AS sentimiento, count(1) AS numTweets

FROM tweets

GROUP BY sentimiento(_3)
```

Ilustración 163 - UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

persona	numTweets
segunda	211
otro	2870
primera	363

Ilustración 164 - UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

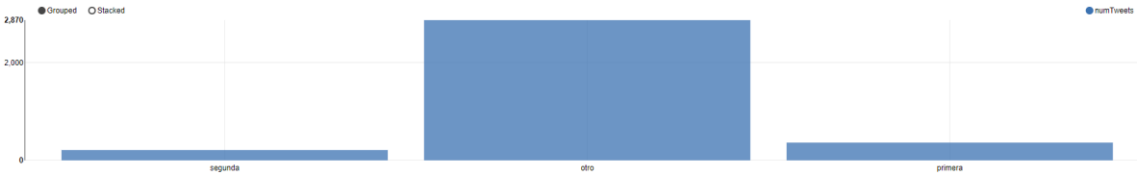


Ilustración 165 - UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

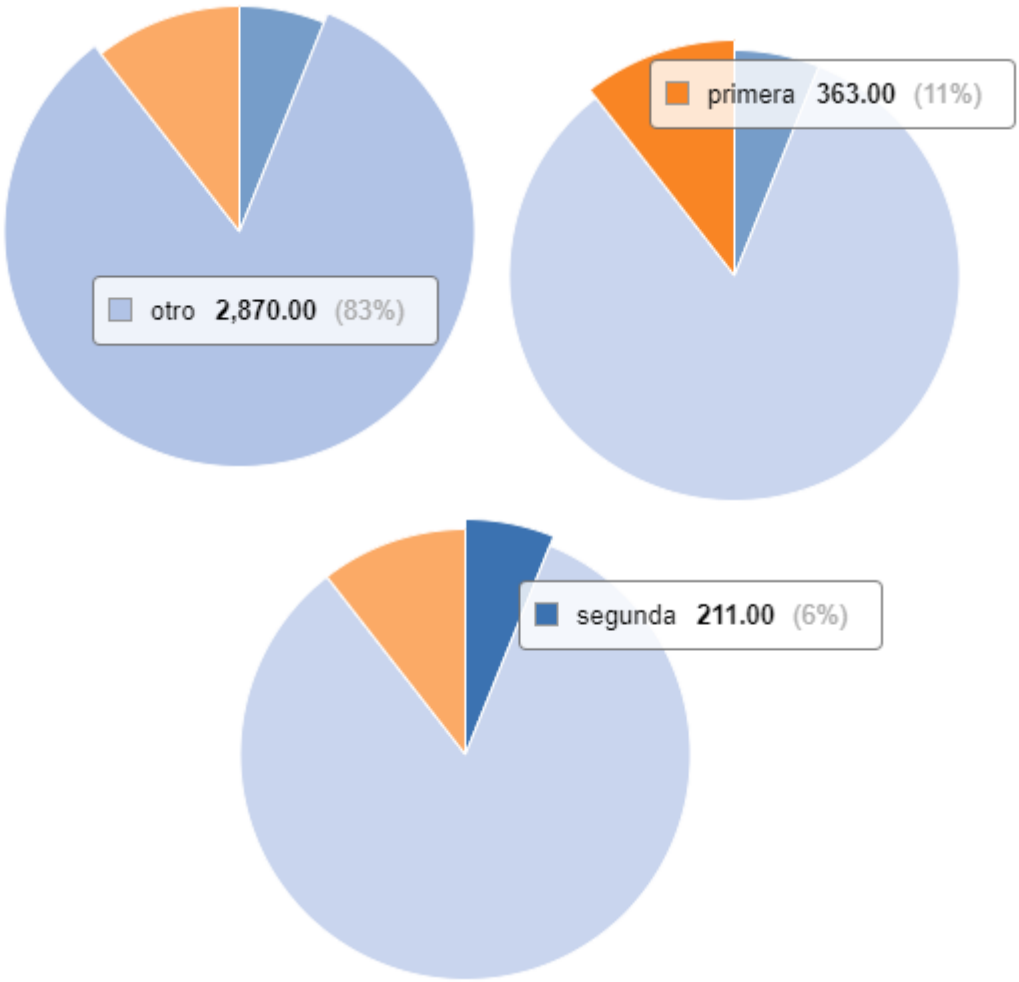


Ilustración 166 - UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

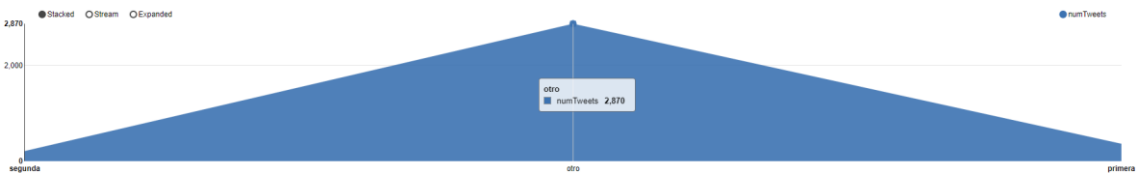


Ilustración 167 - UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

Otra de las consultas que se podría hacer es por ejemplo visualizar los datos de únicamente uno de los tipos. Como analista que estoy analizando los datos, me puede interesar sacar métricas de los que están escritos en primera persona ya que pueden contener información personal relevante sobre dichos usuarios.

```
%sql
SELECT _2 AS fecha, persona(_3) AS persona, tipoDeTweet(_3) AS tipo, wordcount(_3) AS numPalabras, _3 AS texto
FROM tweets
WHERE persona(_3) == "primera"
ORDER BY  _2
```

Ilustración 168 - UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

fecha	persona	tipo	numPalabras	texto
Fri Jun 01 11:28:01 +0000 2018	primera	Original tweet	12	Quedan 183 d\u00edas y es lo que me preocupa \ud83d\ude0c
Fri Jun 01 11:28:02 +0000 2018	primera	RT	21	RT @carloshuitron3: Mitad de a\u00f1o y no he hecho la mitad de las cosas que pensaba hacer este a\u00f1o
Fri Jun 01 11:28:06 +0000 2018	primera	Original tweet	7	me estoy cagando de fr\u00edo ayud
Fri Jun 01 11:28:06 +0000 2018	primera	RT	28	RT @navarrowolff: He luchado toda mi vida contra el clientelismo y la pol\u00edtica tradicional.\nToda ella est\u00e1 hoy con Duque\nYo no me rindo ni\u2026
Fri Jun 01 11:28:47 +0000 2018	primera	RT	30	RT @ahlisto: - Se incendia el barco. Ay\u00fadenos a apagar el fuego o nos morimos todos.\n- Tan agresivo. Ni pide el favor. Pues yo ver\u00e9 si ayud\u2026
Fri Jun 01 11:28:52 +0000 2018	primera	RT	28	RT @Jeancookie23: Vamos Armys! Nos quedan 2 d\u00edas para votar! Ya hemos hecho tanto que pongamos ganas ahora que se termina! \nRT para m\u00e1s
Fri Jun 01 11:28:52 +0000 2018	primera	RT	26	RT @YamilRex: @Sergecast As\u00ed me pas\u00f3 con mi ex... su amigo siempre la recog\u00eda, despu\u00e9s se re\u00eda de m\u00ed.
Fri Jun 01 11:28:55 +0000 2018	primera	Original tweet	12	Estoy despierta hace 1h y no me puedo levantar todav\u00eda....

Ilustración 169 - UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

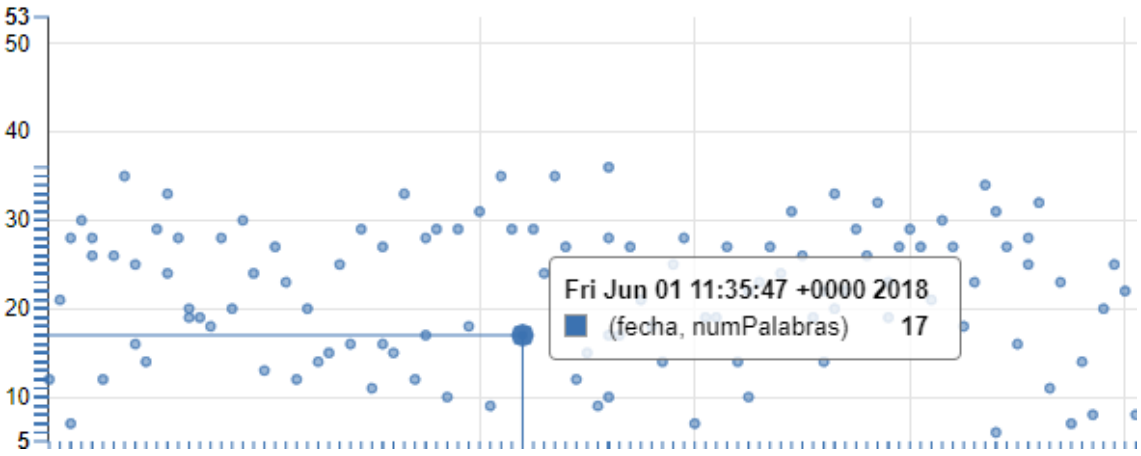


Ilustración 170 - UTILIZACIÓN DE LA FUNCIÓN PRIMERA O SEGUNDA PERSONA

CREAR FUNCIÓN PARA DETECTAR TWEETS POLÍTICOS

Cuando se realizó la ingesta de datos con Apache Nifi accediendo a la API de Twitter, en España se estaba produciendo la moción de censura en contra del presidente del gobierno. Ante esta situación, se decidió que era buena idea analizar cuántos de los tweets que se estaban ingestando estaban relacionados con ese tema. Para ello se ha desarrollado una función en Scala que recoge las palabras clave para detectar este tipo de tweets.

```
def politico(s:String) : String = {

   if(s.contains("rajoy") || s.contains("sanchez")|| s.contains("Rajoy")|| s.contains("Sanchez") ||
    s.contains("gobierno") ||  s.contains("mocion") || s.contains("censura") || s.contains("Mocion")
    || s.contains("Censura") || s.contains("Gobierno") || s.contains("mariano") ||  s.contains("Mariano")
    || s.contains("brey") || s.contains("Brey") || s.contains("pedro") || s.contains("presidente") ||
     s.contains("Presidente") || s.contains("Pablo") || s.contains("pablo") ||  s.contains("Iglesias")
      || s.contains("iglesias") ||  s.contains("congreso") ||  s.contains("Congreso") ||  s.contains("diputado")
      ||  s.contains("Diputado") || s.contains("vota") || s.contains("Vota") ||  s.contains("PNV") ||  s.contains("PP")
      ||  s.contains("Psoe") ||  s.contains("PSOE") || s.contains("Albert Rivera"))
    "Politico"
    else
    "No politico"

}


sqlc.udf.register("politico", politico _)
```

Ilustración 171 - CREAR FUNCIÓN PARA DETECTAR TWEETS POLÍTICOS

UTILIZAR FUNCIÓN TWEETS POLÍTICOS

Una vez que se tiene implementada la función en Scala, se podrá utilizarla en cualquiera de nuestras consultas. En esta primera consulta, se va a realizar la comparación entre los tweets que están hablando de política y los que están relacionados con otro tipo de asunto. Se puede observar en los resultados que de todos los datos ingestados en habla hispana, el 16% de los tweets están hablando sobre la moción de censura.

```
%sql

SELECT  politico(_3) AS tipo, count(1) as numberOfTweets

FROM tweets

GROUP BY politico(_3)
```

Ilustración 172 - UTILIZAR FUNCIÓN TWEETS POLÍTICOS

tipo	numberOfTweets
Politico	540
No politico	2904

Ilustración 173 - UTILIZAR FUNCIÓN TWEETS POLÍTICOS

Ilustración 174 - UTILIZAR FUNCIÓN TWEETS POLÍTICOS

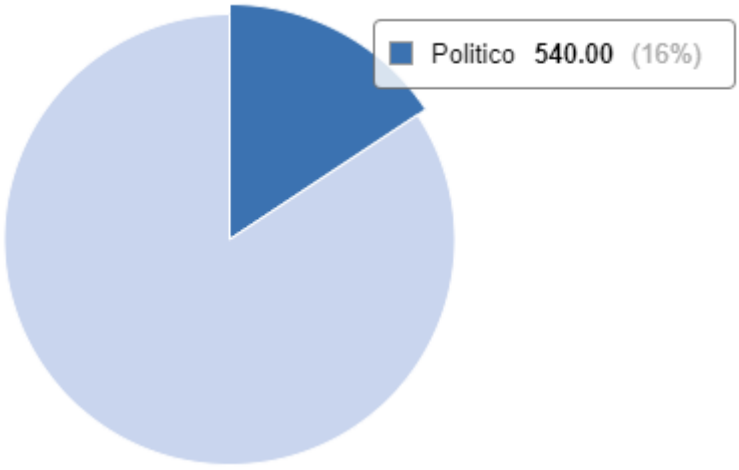


Ilustración 175 - UTILIZAR FUNCIÓN TWEETS POLÍTICOS

En esta segunda consulta que se realiza, se realizan todos los tweets relacionados con la moción de censura y además se mostrará si es contenido nuevo o se trata de un retweet. Se ve que solo un 22% es contenido nuevo, el resto se trata de retweets.

```
%sql

SELECT tipoDeTweet(_3) AS tipo, _3 AS texto

FROM tweets

WHERE politico(_3) == "Politico"
```

Ilustración 176 - UTILIZAR FUNCIÓN TWEETS POLÍTICOS

tipo	texto
RT	RT @marianorajoy: Orgulloso de haber sido vuestro Presidente. Hice todo lo posible y todo lo necesario para dejar las cosas mejor que las e\u2026
Original tweet	Que cambios Propone el Sanchez, ahora que es el Presidente de esta Mierda de Nacion...\n\nAhora que nos Hundira mase\u2026 https:\/\/t.co\/O5K8339e5E
Original tweet	@ManuFernd @Tonicanto1 Porque tu partido se ha quedado solo votando no junto al PP? Al menos podr\u00eda haberse abstenido como los canarios.
Original tweet	Pero vosotros hab\u00e9is votado igualmente NO a la moci\u00f3n de censura. Os hab\u00e9is lucido https:\/\/t.co\/q0wyuEgQX3
RT	RT @diariARA: #\u00daLTIMAHORA Torra es querella per prevaricaci\u00f3 contra Rajoy i Santamar\u00eda https:\/\/t.co\/CQ54QkPbqt
Original tweet	@elnacionalcat #MocionCensura\nAhora ya no ve espa\u00f1oles...Ve racistas... Atenci\u00f3n\nEspa\u00f1oles si algui\u00e9n ha intentado\u2026 https:\/\/t.co\/A9imDQ6lxW
RT	RT @kristynaote: @europapress Home claro! Este Iglesias tiene que garantizarse el sueldo el m\u00e1ximo de tiempo, o quien le va a pagar la hipo\u2026
Original tweet	Oye, esos 580 millones de Euros que el PP le pag\u00f3 al PNV hace 10 d\u00edas...son la prueba de lo cojonudamente que gest\u2026 https:\/\/t.co\/AZRsufY9pq
RT	RT @marianorajoy: El Sr. S\u00e1nchez no est\u00e1 en condiciones de formar un Gobierno estable; no tiene una idea de pa\u00eds; no tiene respuestas a los\u2026

Ilustración 177 - UTILIZAR FUNCIÓN TWEETS POLÍTICOS

Ilustración 178 - UTILIZAR FUNCIÓN TWEETS POLÍTICOS

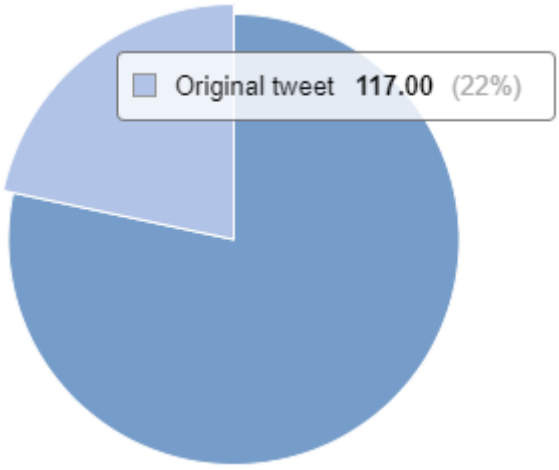


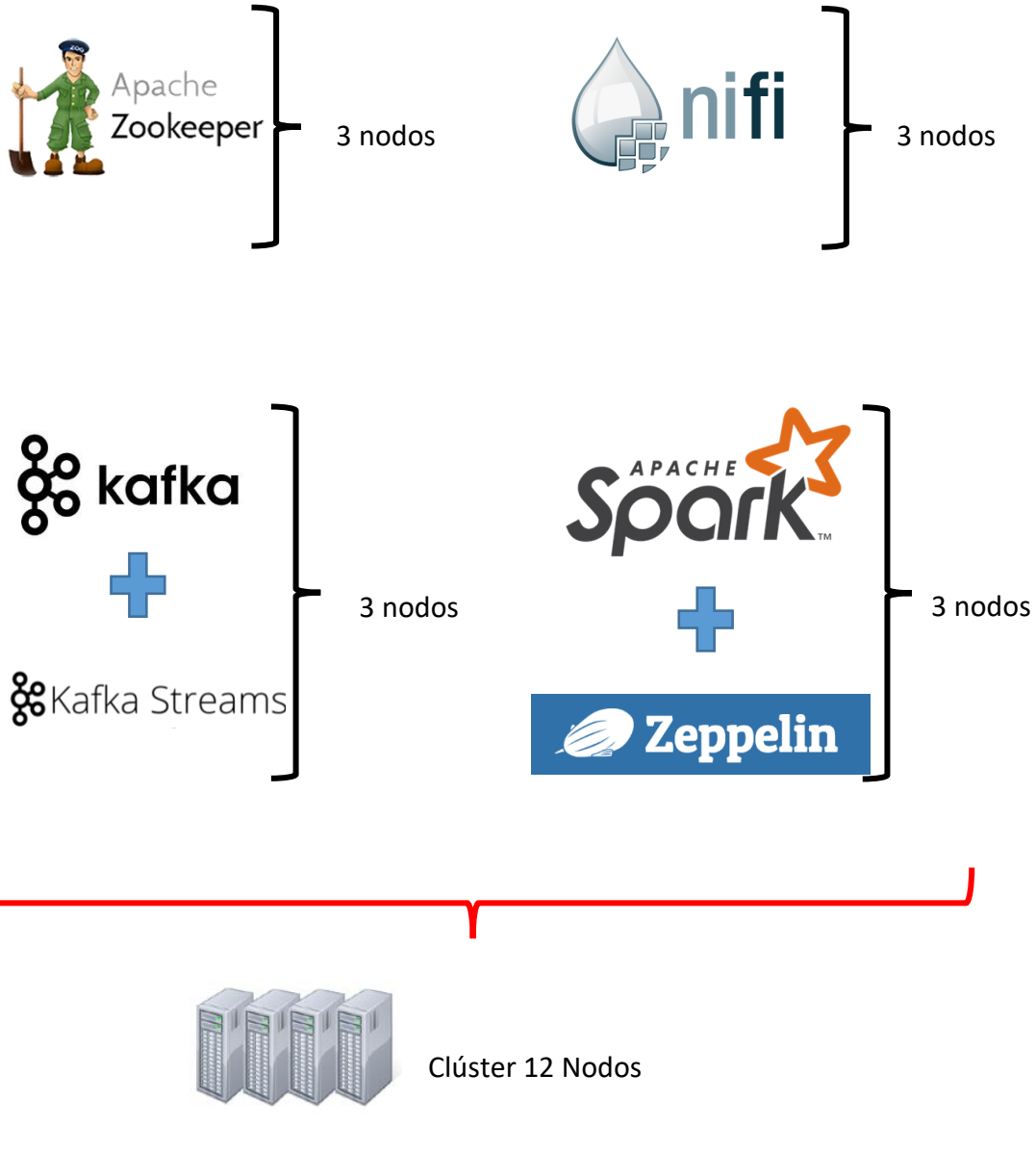
Ilustración 179 - UTILIZAR FUNCIÓN TWEETS POLÍTICOS

PRESUPUESTO

Toda la parte práctica de este TFG se ha realizado en mi ordenador personal, las herramientas utilizadas han sido open source y bajo mi licencia de Windows. Por lo que el cálculo del presupuesto de la construcción, implementación y configuración de esta arquitectura Big Data que se ha realizado sería relativamente sencillo y no aportaría demasiada información útil.

Por este motivo se ha realizado el estudio de qué recursos serían necesarios para la implementación de trasladar esta arquitectura Big Data a un entorno Bid Data real y clusterizado con alta disponibilidad.

La distribución de las herramientas en los diferentes nodos del clúster para su correcto funcionamiento y asegurar que dispone de alta disponibilidad, algo vital en los entornos reales de Big Data es el siguiente:



Una vez vista las necesidades en cuanto a los recursos físicos que se necesitarían para disponer de un entorno Big Data de alta disponibilidad, se va a realizar una tabla para sacar una estimación del presupuesto total sumándole todos los elementos necesarios, teniendo en cuenta que todas las herramientas utilizadas, Apache Nifi, Apache Kafka, Kafka Streams y Apache Zeppelin son Open Source.

Recurso	Unidades	Precio/Unidad	Precio total
Dell PowerEdge T630 Servidor Intel Xeon E5-2620V4/16GB/300GB	12	2.256,59 €	27.079,08 €
Microsoft Windows Server Essential 2016	12	393 €	4.716 €
Developer Big Data	500 h	20 €/h	10.000 €
		TOTAL (IVA incluido)	41.795,08 €

Tabla 1 - Presupuesto proyecto con clúster alta disponibilidad

CONCLUSIONES Y PRINCIPALES DIFICULTADES

Una vez llegados al final de este proyecto se debe mirar atrás y revisar los objetivos que se habían propuesto. El objetivo principal era la creación de nuestro propio entorno Big Data con una arquitectura de principio a fin, que fuese capaz de llegar desde la ingesta de datos hasta la visualización para una correcta explotación de los datos. Como se ha podido ver a lo largo del desarrollo de este trabajo, se ha conseguido que a partir de los datos que facilita la API de Twitter, que son ficheros Json poco entendibles para el usuario final, poder llegar a mostrar de manera visual, concisa y clara, el resultado de consultas con las que el analista puede sacar información y obtener un conocimiento sobre ellos. Pero no sólo se ha logrado eso, sino que, se ha implementado una arquitectura Big data la cual se tiene un control sobre los datos ingestados, tiene un sistema de almacenamiento distribuido, replicado y tolerante a los fallos y finalmente un procesamiento y explotación de los datos en una misma herramienta que facilita el trabajo al usuario.

Hay que tener en cuenta que acceder al API de twitter y hacer alguna consulta de los datos se podría haber hecho en un único procesamiento Spark, pero se quería demostrar lo que es un proyecto de Big Data al completo, ya que el Big Data no es almacenar información o hacer un proceso Spark de unos datos. El Big Data es la ingesta, almacenamiento, procesamiento y explotación, llevando así el tratamiento completo del dato con un control sobre el recorrido del dato.

Se tiene que destacar que al ser un TFG de carácter práctico gran parte del tiempo invertido en el trabajo es de investigación y aprendizaje sobre las distintas herramientas. Ninguna de las herramientas utilizadas las había usado a lo largo de la carrera, lo que ha significado un reto el adentrarme en ellas y tener que desarrollar un trabajo como este. Además me ha llevado a enfrentarme a problemas que no había tenido que hacerlo antes y a buscar posibles soluciones.

- Elección de las diferentes herramientas para formar nuestra arquitectura Big Data. Se ha tenido que ver cuales cumplían mejor con nuestros diferentes casos de uso y además fuesen en consonancia las unas con las otras.

- Todas las herramientas Big Data utilizadas están en plena vigencia y están siendo utilizadas por empresas que tienen proyectos punteros en Big Data.

- Mientras en Nifi y Kafka, existía una gran cantidad información en la documentación oficial tanto de forma teórica como para los desarrolladores Big Data, la documentación de Apache Zeppelin es muy pobre. Lo que ha significado un tiempo extra en el desarrollo con esta herramienta.

- La idea inicial de la arquitectura con la que se inició el desarrollo de este proyecto tuvo que mejorarse ya que se vio que se necesitaba una aplicación para obtener los datos que estaban siendo almacenados en Kafka. Se optó por crear una aplicación en Scala implementando Kafka Streams. Se eligió Kafka Streams para

hacer un seguimiento del dato visualmente a tiempo real, sin esperar a parar a ejecutar antes de ver si va bien y además descubrir una herramienta actual en el mundo del Big Data.

- A la hora de desarrollar en Scala, había la necesidad de utilizar multitud de librerías y por lo tanto descargar numerosos Jar aunque me llevó un tiempo descubrir cómo funcionaba Maven he aprendido los conocimientos necesarios para la utilización de la herramienta que utilizan todos los desarrolladores actuales.

- Comprender la importancia de conocer en cualquier momento el estado y estructura del dato dentro de tu arquitectura para poder realizar el correcto procesamiento.

Como conclusión final, se ha desarrollado un entorno Big Data, el cual ha sido un desafío para mí, ya que desconocía algunos de los conceptos necesarios. También me ha servido para comprender que sin todos los conocimientos adquiridos a lo largo de la carrera hubiese sido imposible afrontar los nuevos conceptos a adquirir y poder concluir este TFG y tengo claro que todos y cada uno de los conocimientos que he adquirido a la hora de realizar este TFG van a ser de gran importancia a la hora de desarrollar mi carrera profesional.

BIBLIOGRAFÍA

- [1] Guller, Mohamed. (2015). Big Data Analytics with Spark. Editorial: Apress
- [2] Holden Karau, Andy Konwinski, Patrick Wendell & Matei Zaharia. Learning Spark. Editorial: O'Reilly
- [3] https://developer.twitter.com/en/docs/basics/getting-started#get-started-app
- [4] https://apps.twitter.com/
- [5] https://developer.twitter.com/
- [6] https://hipertextual.com/archivo/2014/05/que-es-api/
- [7] https://nifi.apache.org/
- [8] https://blogs.apache.org/nifi/
- [9] https://nifi.apache.org/docs.html
- [10] https://nifi.apache.org/developer-guide.html
- [11] https://docs.aws.amazon.com/es_es/kinesisanalytics/latest/dev/about-json-path.html
- [12]https://docs.microsoft.com/es-es/sql/relational-databases/json/format-query-results-as-json-with-for-json-sql-server?view=sql-server-2017
- [13] http://www.javahotchocolate.com/notes/nifi.html
- [14] https://kafka.apache.org/
- [15] https://kafka.apache.org/documentation/
- [16] https://zeent.com/2017/01/13/operaciones-habituales-en-apache-kafka/
- [17] https://bigdatadummy.com/2017/02/01/apache-kafka/
- [18] https://stackoverflow.com/questions/39529511/what-is-the-use-of-consumer-offsets-and-schema-topics-in-kafka
- [19] https://www.tutorialspoint.com/apache_kafka/index.htm
- [20] https://kafka.apache.org/documentation/#connect
- [21] https://kafka.apache.org/documentation/streams/
- [22] http://bigdatums.net/2017/06/22/writing-data-from-apache-kafka-to-text-file/
- [23] https://www.confluent.io/blog/introducing-kafka-streams-stream-processing-made-simple/
- [24] https://dzone.com/articles/self-learning-kafka-streams-with-scala-part-1
- [25] https://es.slideshare.net/GuozhangWang/introduction-to-kafka-streams
- [26] http://zeppelin.apache.org/
- [27] http://zeppelin.apache.org/docs/0.7.3/
- [28] http://zeppelin.apache.org/docs/0.7.3/install/install.html
- [29] https://spark.apache.org/docs/latest/streaming-kafka-0-10-integration.html
- [30] https://spark.apache.org/docs/latest/
- [31]https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.package
- [32] http://xxuan.me/2016-06-28-spark-tutorial.html

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR